МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ЕЛЕКТРОНІКИ, РОБОТОТЕХНІКИ І ТЕХНОЛОГІЙ
МОНІТОРИНГУ ТА ІНТЕРНЕТУ РЕЧЕЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Шутко В.М.
«__»_____ 2021 р.

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА
ЗІ СПЕЦІАЛЬНОСТІ 153 «МІКРО- ТА НАНОСИСТЕМНА ТЕХНІКА»
ОПП «ФІЗИЧНА ТА БІОМЕДИЧНА ЕЛЕКТРОНІКА»

Тема: «Програмовані мультиплексорні наносхеми на базі мажоритарних елементів**»**

Виконавець
студентка групи МН-403Б _____ Тишкова Ірина Олександрівна

Керівник
д.т.н., доцент _____ Мельник Олександр Степанович

Нормоконтролер _____ Сініцин Р.Б.

КИЇВ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: аеронавігації, електроніки та телекомунікацій
Кафедра: електроніки, робототехніки і технологій моніторингу та інтернету речей
Напрям (спеціальність, спеціалізація): 153 «Мікро- та наносистемна техніка»
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
В. М. Шутко

«__»_____2021 р.

ЗАВДАННЯ

на виконання дипломної роботи

Тишкова Ірина Олександрівна
(П.І.Б., випускника)

1. **Тема дипломної роботи:** «Програмовані мультиплексорні наносхеми на базі мажоритарних елементів»

затверджена наказом ректора від « 01» квітня 2021 р. №526/ст

2. **Термін виконання роботи:** з 17 травня 2021 р. по 20 червня 2021 р.

3. **Вихідні дані роботи:** розробити згідно сучасних вимог алгоритми проектування програмованої мультиплексорної наносхеми.
4. **Зміст пояснювальної записки:** перелік умовних скорочень, вступ, класифікація мікро- та ноносхем із програмованою логікою, архітектурні особливості програмованих логічних мікро- та наносхем, програмування мультиплексорів в якості логічних елементів, результатиавтоматизованого програмування мультиплексорних мажоритарних наносхем, висновки, список використаної літератури.
5. **Перелік обов'язкового ілюстративного матеріалу:** графічне представлення результатів автоматизованого програмування мультиплексорних мажоритарних наносхем, схеми, рисунки, таблиці.

1. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Огляд та обробка літератури за темою дипломної роботи	03.04.2021	
2.	Аналіз існуючих рішень	15.04.2021	
3.	Підготовка матеріалів першого розділу кваліфікаційної роботи	01.05.2021	
4.	Підготовка матеріалів другого розділу кваліфікаційної роботи	06.05.2021	
5.	Підготовка матеріалів третього розділу	17.05.2021	
6.	Оформлення документації кваліфікаційної роботи	31.05.2021	

2. Дата видачі завдання: « 02 » квітня 2021 р.

Керівник дипломної роботи _____ Мельник О.С.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Тишкова І.О.
(підпис випусника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Програмовані мультиплексорні наносхеми на базі мажоритарних елементів»: сторінок 74, рисунків 54, таблиці 9, використаних джерел 14.

КВАНТОВІ КОМІРКОВІ АВТОМАТИ, МОДЕЛЮВАННЯ МАЖОРИТАРНИХ ЕЛЕМЕНТІВ, УНІВЕРСАЛЬНІ МАЖОРИТАРНІ ЕЛЕМЕНТА, ТЕРМОГЕНЕРАЦІЯ.

Об’єкт дослідження – Наноелектронна програмована мультиплексорна схема на квантових коміркових автоматах.

Предмет дослідження – Комп’ютерне моделювання та експериментальне проектування наноелектронної схеми з програмованою логікою матриці квантових автоматів.

Мета роботи – розробити комп’ютерну модель проектування наноелектронної схеми програмованої матриці логічних блоків, яка побудована на технології квантових коміркових автоматів та дослідити її експлуатаційні параметри та характеристики.

Метод дослідження – моделювання та автоматизоване проектування програмованих мультиплексорних наносхем в САПР QCADesigner, побудованих на архітектурі квантових коміркових автоматів.

Робота присвячена розгляду проблеми протиріччя між швидкодією та спеціалізацією великих інтегральних схем. Також в роботі розглянуто наносхему на базі універсальних мажоритарних елементів (УМЕ) на квантових коміркових автоматах. Розробка питань теорії і практики використання мажоритарного принципу являється в теперішній час актуальною проблемою, оскільки при наноелектронному виконанні обчислювальних систем з конфігурованими структурами відбувається значне зниження їх вартості і значно спрощується етап автоматизованого системотехнічного проектування. Одна програмована наносхема замінює від 30 до 150 інтегральних схем середнього ступеню інтеграції.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БВВ	блоки вводу-виводу
БМК	базові матричні кристали
ВІС	велика інтегральна схема
ВПБ	вбудовані блоки пам’яті
ГПМЗ	глобальна програмована матриця з’єднань
ДНФ	диз’юктивна нормальна форма
ЕЗЛ	емітерно-зв’язана логіка
ЗП	запам’ятовуючі пристрої
ІС	інтегральна схема
КА	комірковий автомат
КЛБ	конфігуровані логічні блоки
КМОН	комплементарна структура метал-оксид напівпровідник
КНФ	кон’юктивна нормальна форма
ЛЕ	логічні елементи
ЛІЗМОН	транзистор з логічно-ізольованим затвором метал-оксид напівпровідник
ЛК	ланцюг каскадування
ЛПМЗ	локальна програмована матриця з’єднань
МЕ	мажоритарний елемент
МК	макрокомірка
МРТ	матриця розподілу термів
ПБ	перемикаючі блоки
ПКВМ	програмована користувачем вентельна матриця
ПЛІС	програмована логічна інтегральна схема
ПЛМ	програмована логічна матриця
ПМЗ	програмована матриця з’єднань
ПМЛ	програмована матрична логіка
ПНЕС	програмовані наноелектронні системи

ППЗП	перепрограмовані запам’ятовуючі пристрої
ППЗП – ЕС	перепрограмовані запам’ятовуючі пристрої з електричним стиранням
СК	схеми каскадування
СП	схеми перенесення
ТТЛ	транзисторно-транзисторна логіка
УМЕ	універсальний мажоритарний елемент
ФБ	функціональні блоки
ФП	функціональні перетворювачі
CPLD	complex programmable logic devices, складні програмовані логічні пристрої
FPGA	field programmable gate arrays, програмовані користувачем вентильні матриці
SOC	system on chip, програмовані системи на кристалі
SPLD	complex programmable logic devices, прості програмовані логічні пристрої

ВСТУП

Незамінною елементною базою в телекомунікації та сферах цифрової обробки сигналів є ПЛІС, в наслідок змоги формування в них швидкісного досягання алгоритмів в конвеєрному режимі. Виконати таке можливо архітектурою програмовано логічних інтегральних схем, яка має велику множину конвеєрних блоків множення з додаванням, пам'яті. Але побудова обчислювачів на базі ПЛІС зостається трудомісткою, високоякісною роботою.

Стимулом приходу ери нанотехнології був високий темп розвитку мікроелектроніки. Досягнення у науці нанотехнологій в першу чергу виважується на перспективи їх застосування в засобах зв'язку, комп'ютерній техніці, електроніці. Кількість компонентів в блоках динамічної пам'яті та мікропроцесорах на протязі останніх 50 років збільшується в 2 рази кожного півріччя. Завдяки цьому неухильно зменшуються функціональні розміри компонентів, приріст швидкодії, спад енергоспоживання і ціни[7,8,13,14].

Спеціалізація та швидкодія ВІС постійно вступають в суперечку з їх комплексністю, збільшуючи їх сортимент і підвищуючи ціни на електронні пристрої. За допомогою виготовлення конфігурованих наноелектронних приладів можна усунути суперечку між швидкодією і спеціалізацією, розробник за необхідності може змінювати алгоритм роботи визначеної обчислювальної техніки, тобто за допомогою вироблення логічно-арифметичних схем з програмованими особливостями. Під час виготовлення таких схем застосовується цілісний нанотехнологічний комплекс, отже, з позиції інженера це - універсальна робота. Налаштовує вказаний алгоритм роботи мікро- чи ноносхем виконавець виробу,отже, для нього ці схеми виконують вузькоспеціалізовані задачі. Здобуття заданих особливостей можливе за допомогою внесення програмуванням змін будови схеми.

Програмовість - не змога виконати алгоритм обробки кодів, які входять, за допомогою змін програми роботи, як це робить мікропроцесор, а можливість забезпечити виконання на апаратному рівні логічних функцій, за допомогою заміни складової будови ІС[13].

З вище зазначеного, першість програмовано логічних інтегральних схем перед спеціалізованими великими інтегральними схемами - короткий час вироблення ІС з

попередньо зазначеними характеристиками. За такої умови береться типова мікро - або наносхема і подаючи на деякі входи спеціальні сигнали, змінюючи її характеристики. Ця першість встановлює основну ціль таких ПЛІС - заміну груп логічних ІС середнього і великого ступенів інтеграції.

За допомогою крайності апаратного сегменту ПЛІС стає можливе їх програмування, інакше кажучи необхідно встановлення допоміжних виходів і структур налаштування, добавляння ланцюгів інформації і т. д. Споживання програмовано логічних інтегральних схем, їх швидкодія та інші властивості будуть постійно гірші, ніж у приладів на спеціалізованих ВІС. Не дивлячись на це, властивості будуть ліпші за такі само властивості апаратури, яка побудована на типових ІС малого і середнього ступенів інтеграції. Погіршення властивостей є зрівноважуванням за набуття багатофункціональності, слугуючої уніфікації номенклатури виробів електронної техніки, що випускається.

Важливою проблемою сьогодення є побудова питань практики та теорії застосування мажоритарного принципу, тому що наноелектронне виконання обчислювальних систем з конфігурованими структурами знижує їх вартость і значно спрощує етап автоматизованого системотехнічного проектування[7]. 10 000 інтегральних схем може замінити одна програмована наносхема.

РОЗДІЛ 1
КЛАСИФІКАЦІЯ МІКРО – ТА НАНОСХЕМ ІЗ ПРОГРАМОВАННОЮ ЛОГІКОЮ

Мікро- та наносхеми, які програмуються користувачем, відкрили нову сторінку в історії теперішньої мікро- та наноелектроніки та обчислювального обладнання. Вони виготовили ВІС, рекомендований для рішення спеціалізованих задач стандартними виробами електронної промисловості зі всіма випливаючими з цього очікуваними результатами: масове виготовлення, низька ціна мікросхем, скорочення часу виготовлення і виходу на ринок продукції на їх основі. ПЛІС можна класифікувати по багатьом ознакам, в першу чергу це:

- рівень інтеграції і зв’язана з ним логічна складність;
- архітектура (тип функціональних блоків, характер системи їх з'єднань);
- число допустимих циклів програмування;
- тип конфігурації пам'яті;
- ступінь взаємозв’язку між затримкою сигналів та напрямків їх розповсюдження;
- компонентна база (КМОН, ТТЛ, КА та інші);
- однорідність чи гібридність[7,8].

Вище зазначені ознаки віддзеркалюють певну сторону можливих класифікацій, а також мають значення. Виділяючи основні характеристики і їх збільшення, розглянемо групування по трьом, в тому числі двома комплексними, характеристиками:

- архітектурі;
- інтеграційному ступеню і однорідності гібридності;
- числу припустимих рядів програмування і зв’язаному з ним типу конфігурації пам’яті.

В загальному групування ПЛІС виглядає так (рис. 1.1.): CPLD – складні програмовані логічні пристрої; FPGA – програмована користувачем вентильна матриця; FLEX - матриця елементів гнучкої логіки; SOC - система на кристалі[7,8,13].

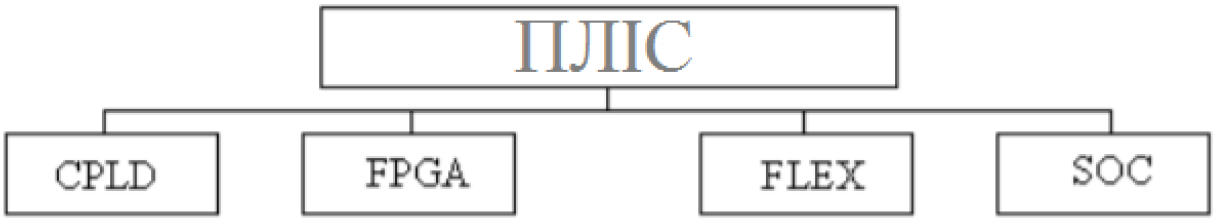


Рис.1.1. Класифікація ПЛІС

CPLD сьогодення, здебільшого, можна електрично перепрограмувати, також вони можуть зберігати логічну структуру після відключення енергопостачання. До складу з CPLD входять блоки логічних вентилів, які об'єднані програмованою комутаційною матрицею. Загальна архітектура інтегральної схеми типу CPLD зображена на рис. 1.2.

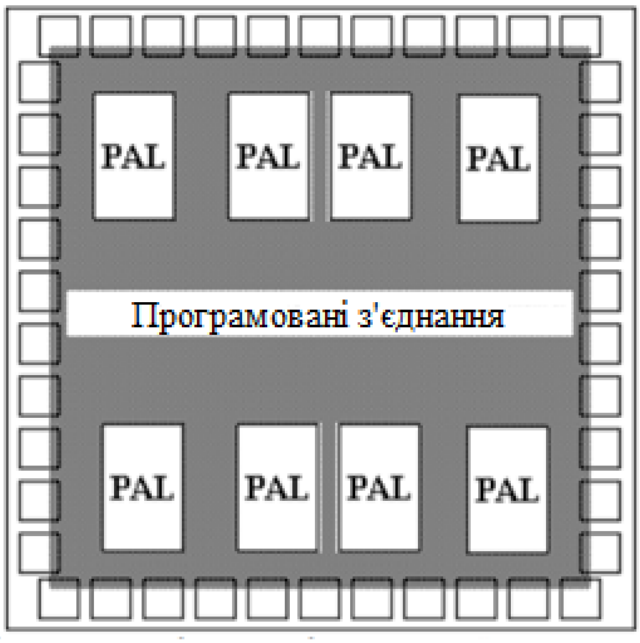


Рис.1.2. Загальна архітектура інтегральної схеми типу CPLD

До цього класу належать ПЛІС сімейств, MAX7000, CoolRanner фірми Xilinx, MAX9000 фірми ALTERA, XC9500, ПЛІС XC7000, MAX5000, Vantis, Lucent, Atmel та інші.

Архітектура CPLD дозволяє легко реалізувати функції , які задаються у вигляді досконалих диз’юктивних нормальних форм, що є дуже привабливим для реалізації цифрових автоматів. Вони є незамінними під час заміни складних схем, реалізованих на звичайних логічних елементах.

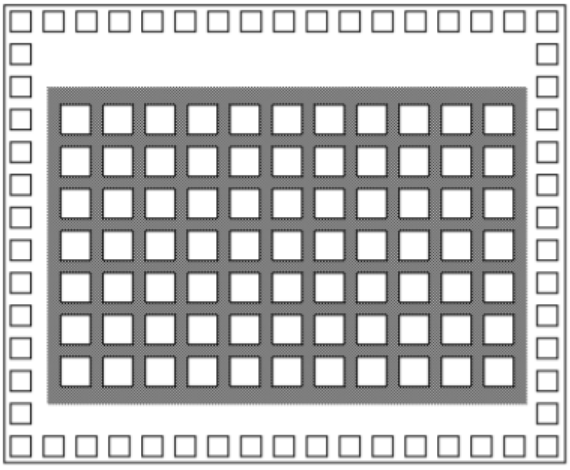


Рис.1.3. Загальна архітектура інтегральної схеми типу FPGA

FPGA (рис. 1.3) – це напівпровідниковий пристрій, який може налаштувати виробник або розробник після виготовлення; з цього й випливає назва: «програмується користувачем». ПКВМ програмуються за допомогою вихідного коду мовою проектування (типу VHDL, на якому можна описати цю логіку роботи мікросхеми), шляхом зміни логіки роботи принципової схеми.

До складу пристрою, що програмується користувачем входить велике число конфігурованих логічних блоків (КЛБ), які розташовані по стовбцям і рядкам у вигляді матриці і трасованих ресурсів.

ПКВМ не залежить від конкретного користувача коли надходить у розпорядження замовника, хоча має уже готові стандартні і не запрограмовані, трасовані ресурси. Як на базі інших програмовано логічних інтегральних схем, так і на основі FPGA, виконується за допомогою впливу на їх запрограмовані з’єднання між собою, завдяки чомуможливо забезпечити замкнутий стан одних ділянок і розімкнутий для інших. При цьому звертатися до виробника ПКВМ немає потреби.

До групи FPGA відносяться деякі ПЛІС фірми Atmel і Vantis, ПЛІС XC5000, XC3000, XC4000, ACT1, ACT2 фірми Actel Spartan, Virtex фірми Xilinx.

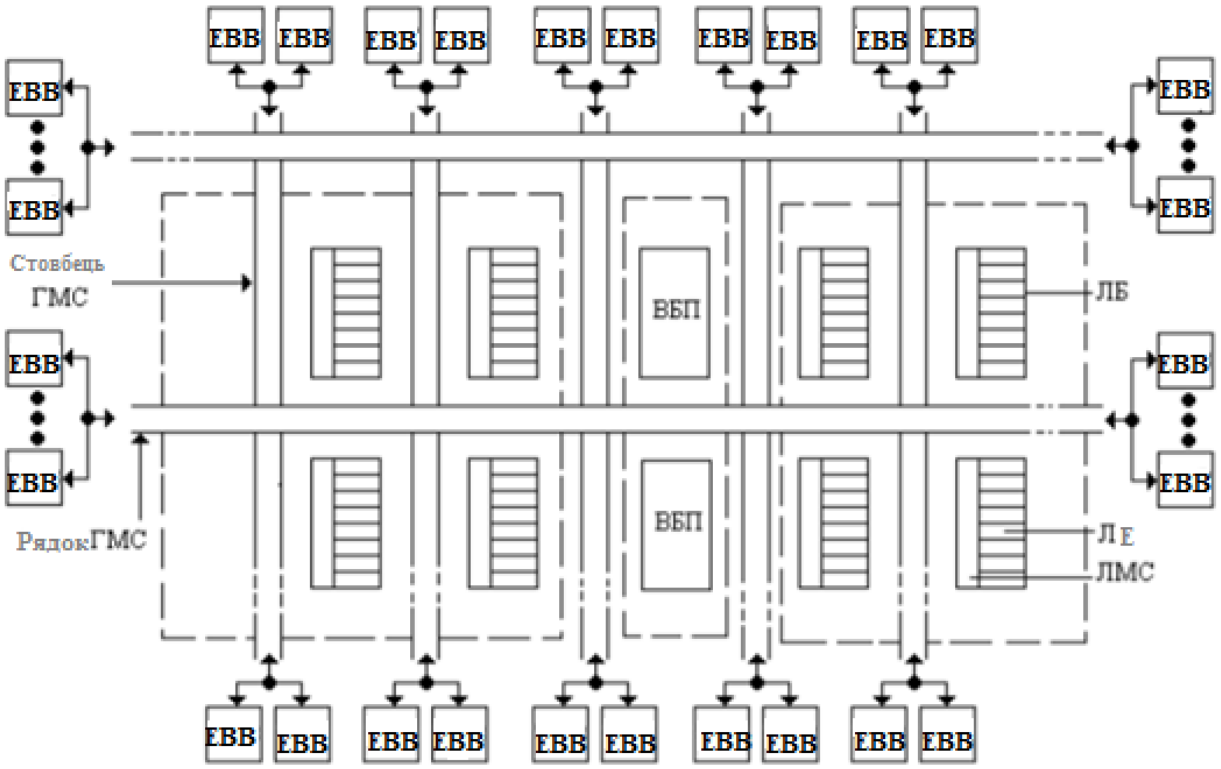


Рис.1.4. Архітектура ПЛІС родини FLEX10K

FLEX - матриця елементів гнучкої логіки. До складу FLEX входять об’єднані в LAB-модулі блоки логічних компонентів по вісім компонентів у кожному. Матриця перемикань створена у вигляді каналів трасування. Завдяки цьому, без зменшення характеристик CPLD, досягнуто зменшення площі кристала, який займаний матрицею перемикань. Компоненти, які обслуговують висновки зовні, а саме компоненти вводу-виводу, приєднують до з’єднуючих каналів.

До цієї групи входять ПЛІС FLEX10K фірми Altera, сімейства FLEX8000 FLEX600, MAX9000. На рис. 1.4 зображено функціональну узагальнену програмовано логічну інтегральну схему типу FLEX. До складу такої архітектури входять логічні блоки, які містять в собі локальну матрицю з'єднань і вісім логічних елементів. Безперервну структуру має глобальна матриця з'єднань, яка розділена на стовпці та рядки. В середині рядків знаходяться вбудовані блоки пам'яті. Також вміщує в себе глобальні ланцюги синхронізації, управління введенням-виведенням.

Новий шлях розвитку комбінованих архітектур поєднує між собою реконфігуровані модулі пам'яті із зручністю реалізації алгоритмів ЦОС на базі LUT.

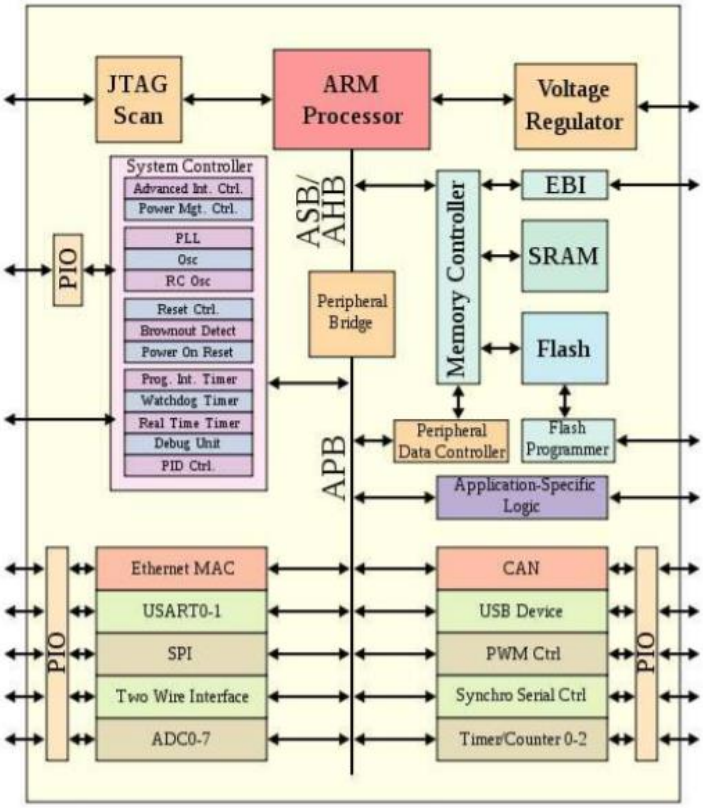


Рис.1.5. Система на кристалі побудована на мікроконтролерах [8].

SOC (рис. 1.5.) — це мікросхема, що вміщує в собі функціональні складові цілого пристрою. В залежності від призначення SoC може оперувати як цифровими сигналами, так і аналоговими, аналого-цифровими, а також частотами радіодіапазону. Типовим застосуванням таких схем є широке різноманіття вбудованих систем.

До складу SOC входить процесор цифрових сигналів або мікропроцесор та мікроконтролер, блок пам'яті, джерело опорної частоти, лічильники та ланцюги затримок після вимкнення, стандартні інтерфейси для зовнішніх пристроїв,таймери. Схеми до складу яких входять більше одного процесора, називаються MPSoC (Multiprocessor System-on-Chip).

Можна виділити 2 підкласи на які ділиться SOC: блочних і однорідних систем на кристалі.

За допомогою програмованості SOC можливо реалізувати різні блоки системи одним і тим же апаратним засобом. Для того що створити різні частини системи потрібно використовувати завчасно реалізовані параметричні функції, так звані “одиниці інтелектуальної власності”.

До складу блочних SOC входять апаратні ядра, тобто спеціалізовані області кристала, які оптимізовують для певної логічної функції, та в яких формуються блоки незмінної структури.

Класифікація обумовлена тіньовою пам'яттю зображено на рис.1.6.

Рис.1.6. Класифікація ПЛІС за ознаками кратності програмування

МОН – транзистори з логічно ізольованими затворами або спеціальні перетинки застосовуються у ПЛІС, які програмуються одноразово. ЛІЗМОН – це транзистор, який має плаваючі затвори, які заряджаються, та які, можуть як заряджатися або розряджатися . Будова з відсутньою можливістю видалення записаних даних застосовується в ЛІЗМОН – транзисторах, які лише частково використовуються для одноразово програмованих ПЛІС.

Плавкі перетинки типу fuse використовувалися у простих ПЛІС перших поколінь, у яких для одержання необхідної конфігурації схеми певна частина перетинок руйнується. Для ПЛІС високого рівня інтеграції схеми з плавкими перетинками, над якими протягом багатьох років велося підбір матеріалів перетинок та удосконалення технологічних процесів, не підійшли.

Перетинки типу antifuse почали застосовуватися в одноразово програмованих FPGA. У начальному стані опір перетинок надиво великий, а в пробитому доволі

малий. Площа перетинок близька до площі перехрестя двох доріжок міжз'єднань - що є дуже компактним.

Одно-затворні ЛІЗМОН – транзистори відіграють роль програмованих елементів у ПЛІС з плаваючим затвором , в яких кристали мікросхем розташовуються в недорогих корпусах , які не мають спеціальних віконець для видалення даних . До програмування транзистора з плаваючим затвором і каналом р-типу затвор не має заряду, і транзистор закритий . Утворенням в транзисторі каналу, який проводить заряд слугує введення в затвор заряду електронів. Протягом кількох десятків років заряд може зберігатися в плаваючому затворі. EPROM - OTP – це пам'ять конфігурації з компонентами вище зазначеного типу. ЛІЗМОН-транзистори мають низьку вартість та малі розміри.

На відміну від попереднього варіанту, мікро- та наносхеми, які можна багаторазово програмувати, відрізняються наявністю засобів видалення записаних в пам'яті даних.

ППЗП-УФ (перепрограмовані запам'ятовуючі пристрої з ультрафіолетовим стиранням) – це пристрої, в яких за допомогою опромінення кристалу ультрафіолетовими променями, дані в компонентах пам'яті типу EPROM видаляються. Кристал з якого видалили конфігурацію можливо знову запрограмувати. Оскільки, ультрафіолетові промені поступово міняють властивості кристалу, то кількість циклів перепрограмування обмежена до сотень або десятків. Саме стирання інформації займає кілька десятків хвилин.

Видалення старої інформації перепрограмованих ПЛІС з конфігурацією пам'яті типу Electrically Erassable Programmable Read - Only Memory виконується за допомогою електричних сигналів. Для цього застосовуються двухзатворні ЛІЗМОН – транзистори. Завдяки звичайному та плаваючому затворам в транзисторі проводиться процес управління. На зовнішніх виходах встановлюються значення програмуючих напруг транзистора, завдяки цьому утворюються режими заряду та розряду плаваючих затворів. ППЗП – ЕС – це пам'ять типу EEPROM. Кількість циклів перепрограмування значно більша за кількість для пам'яті з ультрафіолетовим стиранням даних, і становить 10^5 - $10^{6.}$ Близька до пам'яті типу EEPROM пам'ять конфігурації типу Flash . ЛІЗМОН – це запам'ятовуючі елементи до складу яких

входять два затвори із електричним видаленням даних, які застосовуються для обох видів пам'яті. Різність обох видів пам'яті в організації процесів запису і видалення інформації, також, при виготовлені Flash -пам'яті досягнутий особливо високий рівень параметрів інтеграції , надійності , швидкодії. Розробка Flash -пам'яті є кінцевим пунктом 10-річного розвитку пам'яті типу EEPROM. Обидва види пам'яті з електричним стиранням інформації застосовуються у новітніх ПЛІС.

За другою характеристикою класифікації останній клас ПЛІС - оперативно перепрограмовані. У цих ПЛІС конфігурація задається за допомогою операцій, які без будь-якого спеціального характеру . Великим плюсом попередніх варіантів є те, що для програмування не потрібні ні спеціальні режими з підвищеними напругами і часу впливів на елементи пам'яті, ні спеціальні програматори. Конфігурація пам’яті – SRAM, тобто звичайна тригерна. Властивим завантаженням пам'яті статичної тригерної пам'яті є висока швидкістю, з послідовним потоком бітів або байтів. МОН – транзистор, який керується тригером служить компонентом з програмованою провідністю (режимом "замкнуто - розімкнуто"). Режим останньому транзистору задає стан тригера. Програмованя з’єднань призводить до встановлення тригера в стан 1 чи 0. Незмінний стан тригер зберігає у працюючому режимі. Шляхом перезавантаження конфігурації пам'яті можливо стирати стару конфігурацію і записувати нову необмежене число разів.

Виходячи з того, що тригерна пам'ять не енергозалежна, то відключення живлення призведе до знищення конфігурації ПЛІС , її можна відновити за допомогою завантаження в тригери тіньової пам'яті файл конфігурації з будь-якої пам'яті, яка є енергожалежною . Таке завантаження конфігурації в ланцюжок тригерів тіньової пам'яті проходить протягом десятків чи сотнів мілісекунд в залежності від об’єму файлу.

1.1.Висновок до першого розділу

Перехід від інтегральних схем малого та середнього ступенів інтеграції до великих та надвеликих мікросхем, призводить до урізноманітнення застосувань ПЛІС в цифровій схемотехніці.

Завдяки швидкому зростанню складності електронних схем потрібно використовувати високоінтегровані інтегральні мікросхеми. Сам користувач або ж конструктор може програмувати ПЛІС. Під час програмування до схеми заносяться оборотні або незворотні (в залежності в можливого подальшого перепрограмування) зміни вивідної структури ПЛІС. Головне призначення ПЛІС - заміна логічних інтегральних схем з малим та середнім ступенями інтеграції [7]. Єдина новітня ПЛІС може функціонально замінити від 5 до 60 і більше, в залежності від складності, інтегральних мікросхем малого і середнього ступенів інтеграції [7].

РОЗДІЛ 2
АРХІТЕКТУРНІ ОСОБЛИВОСТІ ПРОГРАМОВАНИХ ЛОГІЧНИХ МІКРО- ТА НАНОСИСТЕМ

Мікропроцесор, в якому розроблена для споживача програма здійснюється послідовно, операція за операцією, називається мікросхема ПЛІС. Електронна схема, яка складається з тригерів і логіки, застосовується у ПЛІС.

Проект для ПЛІС може бути розроблений, наприклад, у вигляді принципової схеми. Verilog або VHDL - це спеціальні мови опису апаратури.

Графічний і текстовий опис проекту, в будь-якому випадку, реалізує електронно-цифрову схему, яку потім буде вбудовано в ПЛІС.

Зазвичай, мікросхема ПЛІС вміщує в себе:

- конфігуруючі логічні блоки, за допомогою яких можна реалізувати потрібну логічну функцію;
- програмованні електронні зв'язки між конфігуруючими логічними блоками;
- програмовані блоки входу / виходу, які забезпечують зв'язок зовнішнього виведення мікросхеми з внутрішньою логікою[8].

Інакше кажучи це не остаточний список. До складу новітніх ПЛІС часто вбудовують додаткові блоки пам'яті, блоки DSP або множники, PLL і інші компоненти.

Розробник проекту на Verilog / VHDL описує логіку роботи майбутньої мікросхеми, яка може бути представлена схемою або ж текстом. Компілятор, знаючи внутрішній устрій ПЛІС сам намагається розмістити необхідну схему за наявними конфігурованим логічним блокам і намагається поєднати ці блоки за допомогою наявних програмованих електронних зв'язків. У загальному випадку розміщення і трасування зв'язків між логічними блоками в ПЛІС залишається за компілятором.

2.1. Складні програмовані логічні пристрої – CPLD

CPLD – Складні програмовані логічні пристрої, це єдиний вид інтегральної схеми, яку розробники програм розробляють для реалізації цифрового обладнання, такого як мобільні телефони[7,13]. Вони можуть обробляти вищі конструкції, ніж SPLD, але пропонують менше логіки, ніж FPGA. CPLD включають численні логічні

блоки; кожен з блоків включає 8-16 макроелементів. Оскільки кожен логічний блок виконує певну функцію, усі макроелементи логічного блоку повністю зв’язані. Залежно від використання, ці блоки можуть підключатися один до одного, а можуть і не підключатися.

CPLD доступні у декількох формах пакетів IC та логічних сімействах. CPLD також відрізняються щодо напруги живлення, робочого струму, струму в режимі очікування та розсіювання потужності. Крім того, їх можна отримати з різним обсягом пам'яті та різними видами підтримки пам'яті. Зазвичай пам’ять виражається в бітах / мегабітах. Підтримка пам'яті складається з ПЗУ, оперативної та двопортової оперативної пам'яті. Він також включає в себе пам'ять CAM (адресація вмісту), а також пам'ять FIFO (перший вхід, перший вихід) та пам'ять LIFO (останній вхід, останній вихід).

Складний програмований логічний пристрій складається з групи програмованих FB (функціональних блоків). Входи та виходи цих функціональних блоків з'єднані між собою GIM (глобальна матриця взаємозв'язку). Цю матрицю взаємозв'язку можна переконфігурувати, щоб ми могли змінювати контакти між функціональними блоками. Будуть деякі вхідні та вихідні блоки, які дозволять нам об’єднати CPLD із зовнішнім світом. Узагальнена структура CPLD показана на рис. 2.1 [8].

Як правило, програмований FB виглядає як масив логічних шлюзів , де можна запрограмувати масив І, а АБО стабільний. Але кожен виробник має власну думку, щоб розробити функціональний блок.

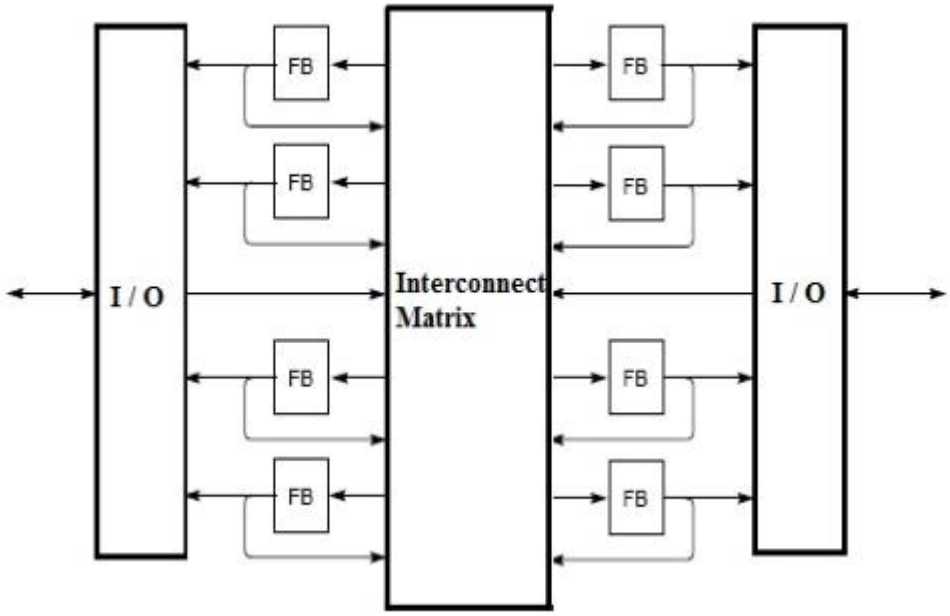
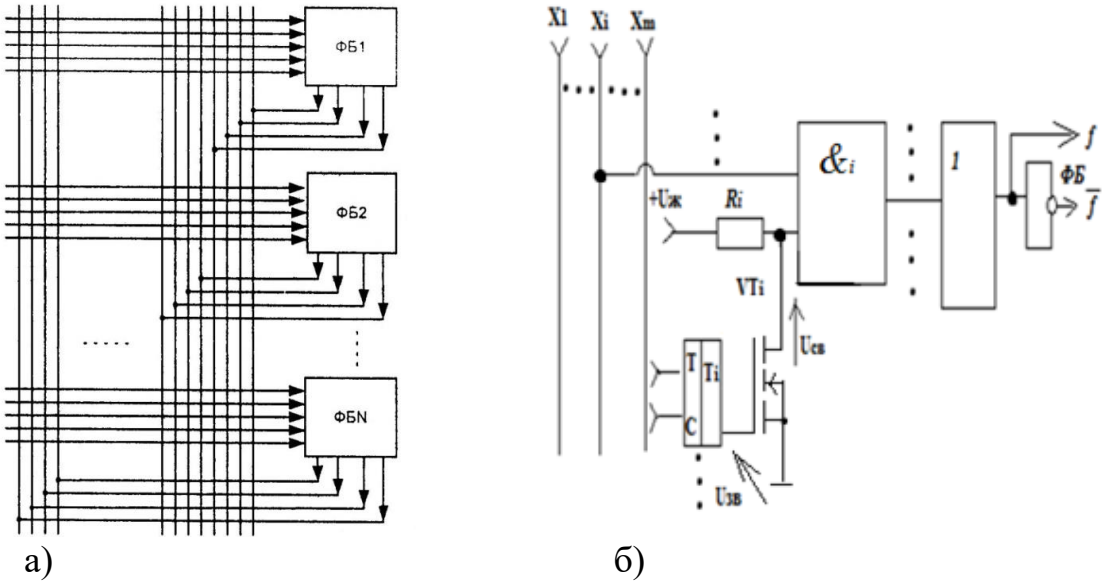


Рис. 2.1 – Узагальнена структура CPLD

I/O пов'язані із зовнішніми двунаправленими виводами які, залежно від програмування, можуть бути використані як входи або як виходи. Три нижніх вивода або спеціалізуються для подачі на матрицю функціональних блоків сигналів GCK (Global Clocks) глобального тактування, сигналів GSR (Global Set - Reset) глобальної установки-скидання і сигналів GTS (Global 3-state Control) глобального управління третім станом вихідних буферів або ці ж виводи можуть бути використані для операцій вводу-виводу. Тут і далі термін глобальний застосовується для сигналів, загальних для всієї мікро- чи наносхеми.

Програмована матриця з'єднань. У ПМЗ (рис. 2.2, а) виходи функціональних блоків ФБ підключаються до вертикальних безперервних (несегментованим) ліній, причому кожному виходу відповідає своя лінія .



а) б)

Рис. 2.2 – Схема ПМЗ CPLD (а) та схема передачі сигналів із цієї матриці в функціональний блок (б)

Входи ФБ пов'язані з горизонтальними лініями, що перетинають всі вертикальні лінії . На перетинах горизонтальних і вертикальних ліній є програмовані точки зв'язку, так що будь-який вхід ФБ може бути підключений до будь-якого виходу, чим забезпечується повна комутованість блоків. вертикальні лінії .

Перевагою ПМЗ розглянутого типу є мала і передбачувана затримка комутованих сигналів, так як для кожного з'єднання створюється ідентичний всім іншим канал зв'язку з малим числом програмованих ключів або навіть їх відсутністю, якщо передача сигналів з ПМЗ в ФБ організована так, як показано на рис. 2.2, б. У цьому випадку відсутні програмовані ключі в колах передач сигналів, а програмуються тільки напруги на нижніх входах кон'юнкторів (логічних елементів І), і ФБ отримає сигнал від і-ї вертикальної лінії ПМЗ (i = 1, 2 , ... , m). Якщо сигналом лог.0 від запрограмованого тригера T_i n-канальний МОН-транзистор VT_i буде закритий ($U_{зв} \approx 0$), то на нижньому вході i-го кон'юнктора діятиме високий потенціал лог.1 ($U_i^1 = U_{св} \approx +U_ж$). Відкритий транзистор VT_i($U_{зв} \approx +U_ж$) підключає нижній вхід кон'юнктора до нульового потенціалу ($U_{св} \approx 0$), створюючи на ньому і на виході кон'юнктора сигнал лог.0. Таким чином задаючи тригеру T_i сигнал логічного нуля, іншим тригерам сигнал логічної одиниці, можна забезпечити закритий стан транзистора T_i і відкритий стан всіх інших транзисторів, що означає

підключення входу ФБ до і-ї вертикальної лінії ПМЗ з утворенням безперервного з'єднання.

Замкнуті транзисторні ключі мають, в першому наближенні, схему заміщення у вигляді інерційного RC-ланцюга і вносять основні затримки у процес поширення сигналу.

Блоки вводу - виводу CPLD. БВВ з'єднують зовнішні контакти мікро- чи наносхеми з її внутрішніми ланцюгами. Характерним прикладом такого блоку може служити БВВ CPLD типу XC9500 фірми Xilinx , показаний на рис. 2.3.

Основою БВВ служать два буфера - вхідний (1) і вихідний (2). Щоб забезпечити постійність рівнів напруги , що надходять на вхідний буфер , і їх незалежність від амплітуди вхідних сигналів , у схемі виробляється внутрішня напруга живлення $U_{жINT}$ і вводиться ланцюг з двох фіксуючих діодів.

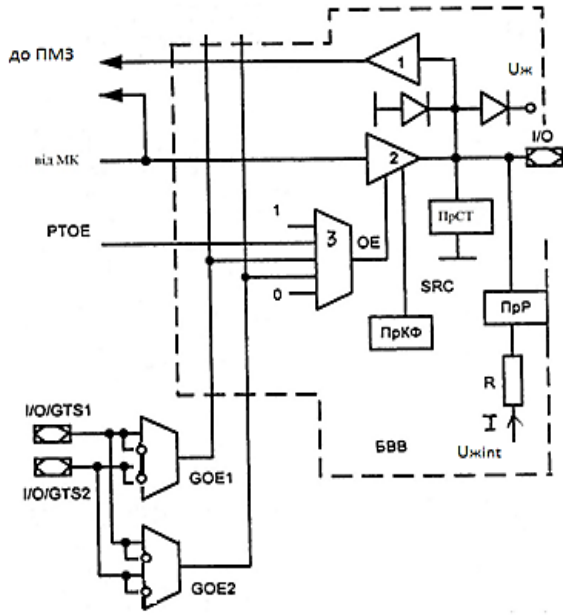


Рис. 2.3. Приклад схеми вводу- виводу CPLD

Схема програмованої спільної точки ПрСТ дозволяє користувачу при необхідності отримувати додатковий « заземлений » вивід. Додаткові виводи для системи « заземлення » підвищують її якість і тим самим знижують рівень завад в мікро- чи наносхемі.

Схема програмування вмикаємого резистора ПрР введена для виключення плаваючих потенціалів на контактах вводу, коли вони не використовуються в робочому режимі. У цьому випадку контакту задається високий потенціал від

ланцюга $U_{жINT} - IR$. Резистор використовується і в деяких інших режимах , а в робочих режимах відключається.

Вихідний буфер 2 отримує сигнали дозволу роботи OE та управління крутизною фронту вихідної напруги SRC (Slew Rate Control) . Сигнал OE за допомогою програмованого мультиплексора 3 виробляється в декількох варіантах: від терма PTOE , одержуваного від макрокомірки, від будь-якого з глобальних сигналів управління третім станом (GOE1 , GOE2), від константи 1 та від константи 0. Глобальні сигнали управління третім станом утворюються з можливістю вибору будь-якої полярності вихідних сигналів GTS1 і GTS2.

Вихідні буфери конфігуруються для роботи з напругою живлення 5 або 3,3 В при підключенні зовнішнього джерела живлення з тим чи іншим рівнем напруги (ці цифри відносяться до блоку типу XC9500, зараз у блоків вводу / виводу рівні вихідних сигналів можуть вибиратись з багатьох значень, в тому числі для наносхем з таких низьких напруг, як 2,5 і 1,8 В).

В клас CPLD потрапляють ПЛІС з рівнем інтеграції 600-100000 еквівалентних елементів, числом макрокомірок 32-512, числом функціональних блоків 2-16 і часом поширення сигналу від будь-якого входу до будь-якого виходу 1-10 нс.

2.2. ПЛІС з комбінованою архітектурою- FLEX

З ростом рівня інтеграції ПЛІС їх архітектури ускладнювалися, з'явилися архітектури, які в тій чи іншій мірі поєднують переваги CPLD і FPGA. До числа мікро- та наносхем з такою архітектурою, в першу чергу, можна віднести сімейство FLEX (Flexible Logic Element matriX). На рис. 2.4 наведена структура мікросхем сімейства FLEX.

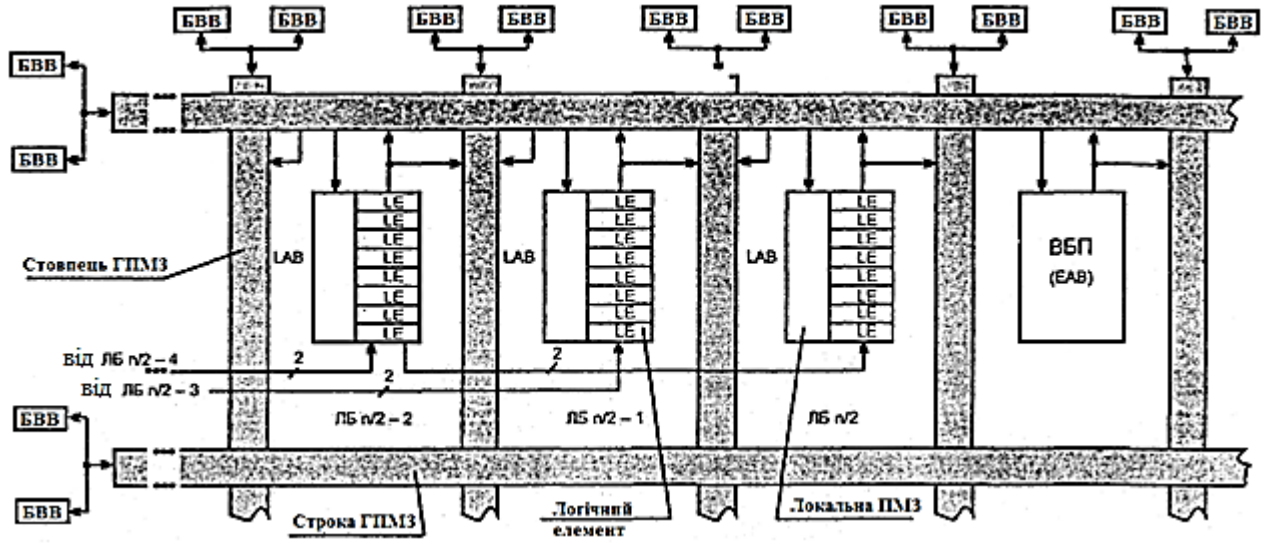


Рис. 2.4. Структура мікро- та наносхем сімейства FLEX

Мікро- та наносхеми сімейства FLEX мають функціональні блоки (LABs, Logic Array Blocks) з логічними елементами ЛЕ (LEs, LogicElements), що містять функціональні перетворювачі ФП табличного типу (LUTs). Функціональні блоки розташовані у вигляді матриці, між їх рядками і стовпцями проходять горизонтальні і вертикальні канали трасування, що характерно для FPGA. У той же час, траси в каналі не сегментовані, а неперервні, що типово для CPLD. Оскільки, як уже зазначалося, в схемах з великим числом функціональних блоків застосування єдиної комутаційної матриці ускладнено, система комутації має два рівні міжз'єднань - глобальний і локальний. Локальна програмована матриця з'єднань (локальна ПМЗ або ЛПМЗ) забезпечує міжз'єднання логічних елементів ЛЕ, з яких складаються функціональні блоки - LABs. До складу LAB входять 8 логічних елементів. З'єднання між блоками LAB забезпечуються глобальною програмованою матрицею з'єднань ГПМЗ, до кінців рядків і стовпців якої підключаються блоки вводу / виводу (IOBs, Input / OutputBlocks).

У складі багатьох ПЛІС з'явилися вбудовані блоки пам'яті ВБП (EABs, Embedded Array Blocks). Раніше існувала можливість використання в якості субмодулів пам'яті лише тих ресурсів, які є в ФП типу LUT.

У схемі на рис. 2.4 показаний включений у середині рядків вбудований блок пам'яті (його ємність дорівнює 2048 біт). Такий блок може конфігуруватися як ЗП з

організацією 256x8, або 512x4, або 1024x2, або 2048x1 і використовуватися не тільки для зберігання даних, але і як табличний ФП для реалізації складних логічних функцій з числом аргументів 8-10 (зокрема, на блоках LAB будуються швидкодіючі арифметико-логічні пристрої (АЛП), помножувачі 4x4 і т.д [8].

Логічні елементи(ЛЕ**)** сімейства FLEX (рис. 2.5) мають у своїй основі 4-входові функціональні перетворювачі ФП табличного типу (LUT). Особливістю схем, які можуть бути побудовані з цих логічних елементів, є наявність спеціальних трактів перенесення, що утворюються з ланцюжків схем перенесення СП, і трактів каскадування, що утворюються зі схем каскадування СК з безпосередніми і швидкодіючими зв'язками між логічними елементами по вказаних трактах.

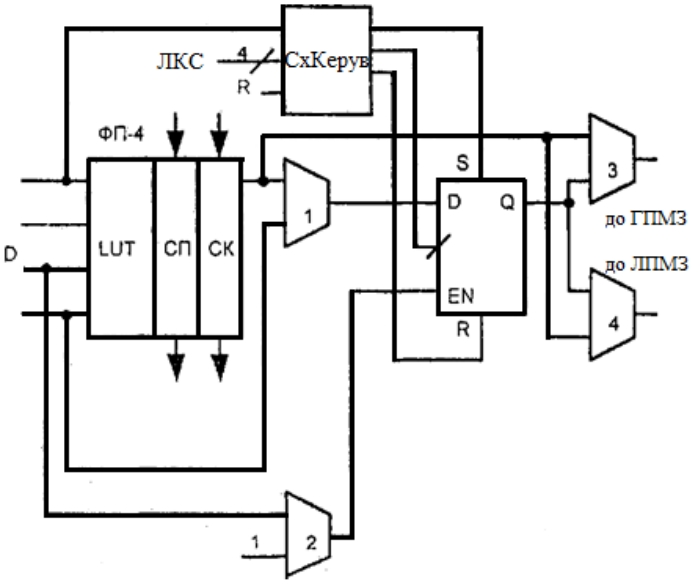


Рис. 2.5. Схема логічного елемента сімейства FLEX

ФП з 4 входами має 16 біт пам'яті і для відтворення логічних функцій 4 аргументів організується у варіанті 16x1. Ті ж самі 16 біт можна використовувати у вигляді двох табличних ФП з організацією 8x1, які реалізують дві функції 3 змінних, що відповідає, наприклад, потребам побудови розрядних схем суматорів з послідовним перенесенням, розрядних схем деяких лічильників і т.д.

У мікросхемах сімейства FLEX затримка ланцюга перенесення є малою (приблизно 1 нс), що робить доцільним застосування в проектах багатьох простих схем з послідовними переносами навіть для швидкодіючих пристроїв. Синхронний тригер у схемі на рис. 2.5 отримує сигнали від схеми керування з входами R (Reset), D (одна з

вхідних ліній логічного елемента) і ЛКС (чотири лінії локальних керуючих сигналів). Один з двох сигналів ЛКС по вибору використовується для тактування тригера, два інших разом з сигналом R і одним з входів D керують режимами скидання / установки. За допомогою програмування можна задавати декілька режимів впливу на тригер по входах S і R. Всі ці режими (скидання, установка, завантаження в різних варіантах) асинхронні.

Тригер може бути використаний для фіксації значень вироблених ФП функцій або як окремий елемент з входом від однієї з ліній D в залежності від програмування мультиплексора в лінії цього входу.

Вихідний сигнал логічного елемента через програмовані мультиплексори 3 та 4 може подаватися в глобальну ГПМЗ і локальну ЛПМЗ матриці міжз'єднань в програмованих варіантах комбінаційного (з обходом тригера) або регістрового (з тригера) виходів.

Ланцюг каскадування ЛК використовується для отримання. функцій з числом аргументів більше 4. Три сусідніх ЛЕ можна застосувати для відтворення часткових функцій, а потім за допомогою каскадування сформувати з цих функцій остаточний результат (рис. 2.7, а). При отриманні з часткових функцій єдиної функції багатьох змінних часткові функції суміжних ЛЕ об'єднуються будь-якою логічною операцією над двома змінними, крім функцій нерівнозначності (додавання по модулю 2) і рівнозначності (на рис. 2.6, а функції позначені умовними значками).

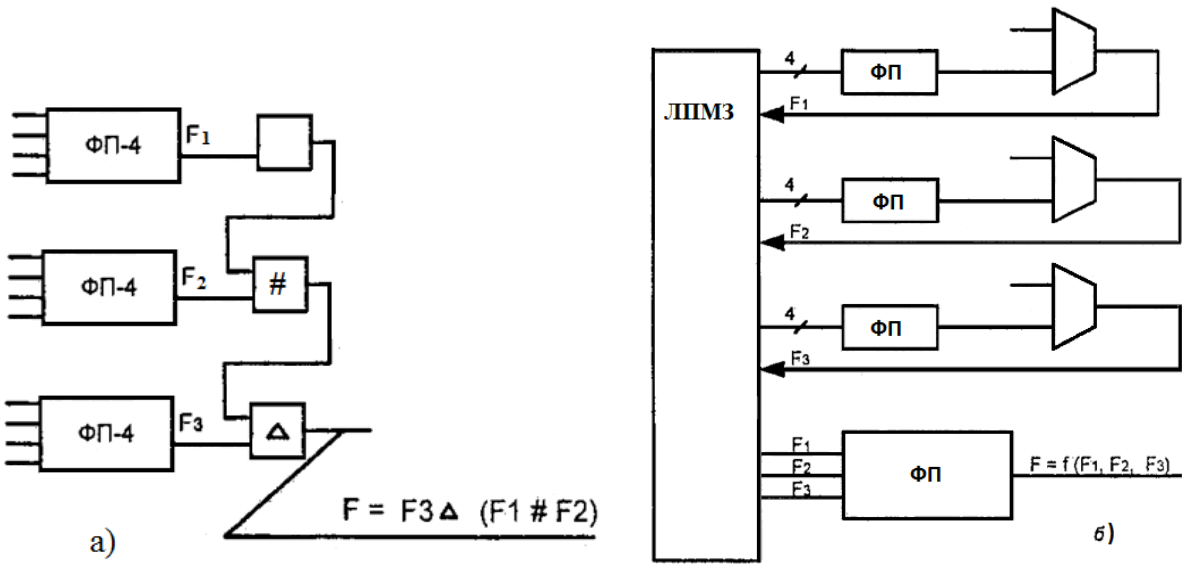


Рис. 2.6. Способи відтворення функцій багатьох змінних методами каскадування (а) та декомпозиції (б)

За допомогою зворотних зв'язків також можна отримати логічні функції багатьох змінних. При цьому спочатку виробляється деяка функція чотирьох змінних, потім вона вводиться в якості одного з входів в інший ЛЕ до локальної матриці програмованих з'єднань ЛПМЗ. В результаті обчислюється "функція від функцій" з числом аргументів, що перевищує 4 (рис. 2.6, б).

Вбудовані блоки пам'яті. До складу сімейства FLEX10K були вперше включені вбудовані блоки пам'яті (ВБП) загальною ємністю приблизно від 6 до 20 Кбіт для різних представників сімейства. Блоки вбудованої пам'яті можна використовувати як реалізації ППЗП і логічних схем, і як статичний ЗП. Один вбудований блок пам'яті при організації 256x8 реалізує помножувач 4x4, здатний працювати на частотах до 50 МГц. Побудова такого ж помножувача на типових ЛБ потребувало б зайняти 8 логічних блоків, а частота роботи помножувача не перевищила б 20 МГц.

У структурі вбудованих блоків пам'яті (рис. 2.7) крім модуля пам'яті RAM/ROM є кілька синхронних D-тригерів і програмованих мультиплексорів 1 ... 7. Локальна програмована матриця з'єднань ЛПМЗ отримує 22-26 сигналів від рядка глобальної матриці ГПМЗ. Регістри (D-тригери) 1 і 2 програмуються для передачі в модуль пам'яті даних і адре́с різної розрядності в залежності від заданої конфігурації пам'яті. У блоці з ємністю 2 Кбіт розрядність даних може змінюватися від 1 до 8, а розрядність адреси від 11 до 8. Запис у пам'ять в залежності від програмування мультиплексорів 4-6 може бути синхронною (від регістрів по сигналах тактування) або асинхронною (безпосередньо від ЛПМЗ).

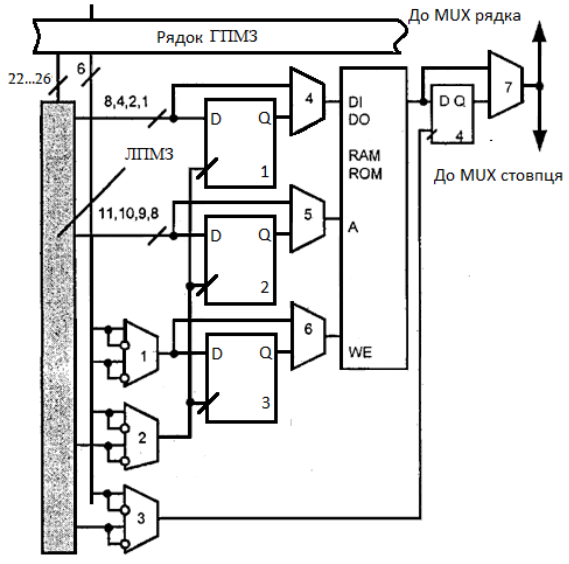


Рис. 2.7. Структура вбудованих блоків пам'яті в мікросхемах сімейства FLEX10K

Сигнали керування регістрами (D-тригерами 1-3) поступають від глобальної шини керуючих сигналів з можливістю вибору їх полярності (мультиплексори 1-3). Вихідні сигнали блоку пам'яті за допомогою мультиплексора 7 можуть передаватися як на лінії рядка, так і на лінії стовпця ГПМЗ в тактованому або асинхронному варіантах.

2.3. Програмовані системи на кристалі – SOC

SoC - це мікросхема, яка інтегрує всі компоненти в єдину мікросхему. Він може містити аналогові, цифрові, змішані сигнали та інші функції радіочастоти, які лежать на одній мікросхемі. Сьогодні SoC дуже поширені в електронній промисловості через низьке енергоспоживання. Крім того, вбудовані системні програми чудово використовують SoC [7,8].

Процес виготовлення SoC можна поділити на чотири етапи:

1. Розробка архітектури SoC на системному рівні.

2. Вибір наявних СФ-блоків з бази даних (всередині фірми, інших фірм або постачальників СФ-блоків).

3. Проектування відсутніх блоків.

4. Інтеграція всіх блоків на кристалі.

Виробник SoC може вибирати такі рішення:

- самостійна розробка необхідних СФ-блоків;
- покупка СФ-блоків у провідних розробників і виробників мікросхем;
- пошук і застосування СФ-блоків, що надаються у відкритому доступі.

SoC складаються з:

- Блок управління: У SoC основними блоками управління є мікропроцесори, мікроконтролери, цифрові процесори сигналів тощо.
- Блоки пам'яті: ПЗУ, оперативна пам'ять. Флеш-пам’ять та EEPROM є основними блоками пам’яті всередині мікросхеми SoC.
- Одиниці синхронізації: Генератори та ФАПЧ є одиницями синхронізації системи на мікросхемі.
- Інші периферійні пристрої SoC - це таймери лічильника, таймери реального

часу та генератори живлення.

- Аналогові інтерфейси, зовнішні інтерфейси, регулятори напруги та блоки управління живленням формують основні інтерфейси SoC.

Загальний план мікросхем зображений на рис. 2.8. У ній поєднуються функціональні блоки типів LUT і ПМЛ. Крім того, мікросхеми мають вбудовані блоки пам'яті ESB (Embedded System Blocks), що містять також засоби для реалізацій ДНФ логічних функцій (що і пояснює термін "системні" в назві блоків).

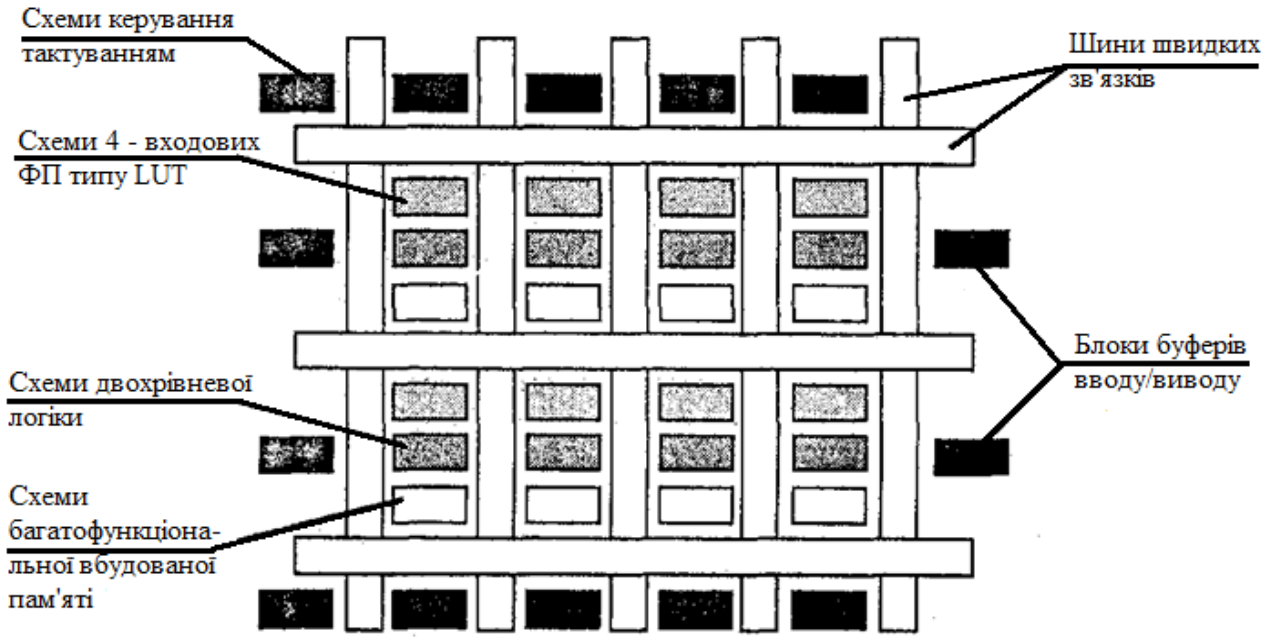


Рис. 2.8. Загальний план кристала мікросхем сімейства APEX20K/KE

СФ-блоки не повинні займати більшу частину кристалу та використовувати багато зовнішніх компонентів, а повинні включати елементи інфраструктури системи живлення (інтерфейси, системи харчування і синхронізації, вбудовані засоби контролю).

На рис.2.9. зображено мікросхему типу Zynq-7000, це один із можливих варіантів складу структури SoC [7,8].

Завдяки незалежним ланцюгам харчування, які входять до складу процесорної системи та програмованої логіки, для зниження енергоспоживання можливо відключати блок логіки. Також можливо відключати невикористовуваних периферійних пристроїв та динамічно управляти тактовою частотою процесорів. Формувачі для розподілу тактових сигналів на синтезатори частоти, схеми фазового

зсуву та високошвидкісні буфери входять до системи управління тактовими сигналами.

Багатофункціональні ОС, такі як Linux або Windows, забезпечують ефективне використання мережевих можливостей і розвиненого призначеного для користувача інтерфейсу. Багатофункціональні ОС дозволяють економити апаратні ресурси, управляти завданнями або процесами, в тому числі в режимі реального часу.

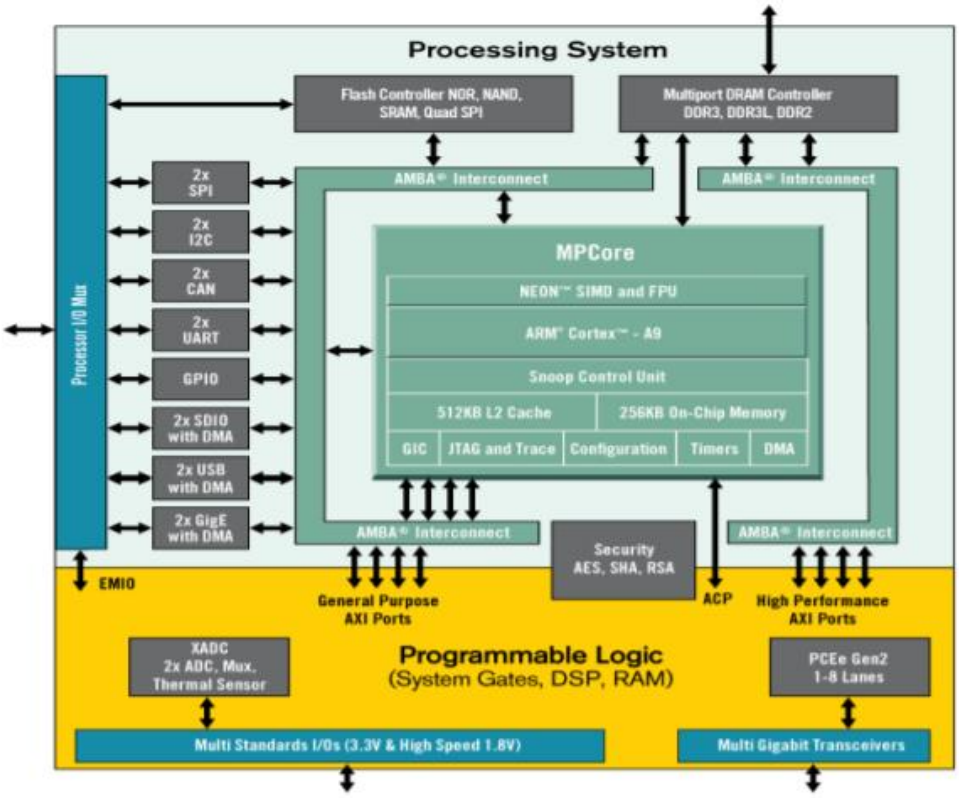


Рис.2.9.Програмований SoC Zynq-7000, що містить 2 ядра сімейства процесорів ARM Cortex-A9

Проектування SoC націлене на розробку апаратного та програмного забезпечення конструкцій SoC. Загалом, проектний потік SoC складається з:

- Апаратні та програмні модулі: Апаратні блоки SoC розроблені з попередньо кваліфікованих апаратних елементів та програмних модулів, інтегрованих із використанням середовища розробки програмного забезпечення. Для розробки модулів використовуються мови опису обладнання, такі як Verilog, VHDL та SystemC.
- Перевірка функціональності: SoC перевіряються на правильність логіки перед передачею ливарному цеху.
- Перевірка апаратних та програмних конструкцій: Для перевірки та

налагодження апаратного та програмного забезпечення конструкцій SoC інженери застосували ПЛІС , прискорення моделювання, емуляцію та інші технології.

- Місце і маршрут: Після налагодження SoC, наступним кроком є розміщення та направлення всієї конструкції до інтегральної схеми, перш ніж вона буде передана на виготовлення. У процесі виготовлення зазвичай використовуються повнофункціональні, стандартні технології комірок та FPGA.

Переваги SoC

- Низька потужність.
- Низька вартість.
- Висока надійність.
- Малий форм-фактор.
- Високий рівень інтеграції.
- Швидка робота.
- Більший дизайн.
- Маленький розмір[8].

Недоліки SoC

- Вартість виготовлення.
- Підвищена складність.
- Час до вимог ринку.
- Більше перевірки.

Різновиди SoC

NVIDIA Tegra 3 - це платформа сімейства Tegra, яка використовується в різних пристроях Android. Деякі пристрої, такі як Asus Eee Pad, HTC One X та Google Nexus Tablet, використовують Tegra 3 на платі. Це поставляється з центральним процесором і п’ятьма ядрами. Кожне ядро - це мікросхема Cortex A9 ARM, тоді як п'яте ядро виконане з використанням кремнію з низькою потужністю і має швидкість 500 МГц.

Qualcomm важлива, коли використовуються смартфони та планшети Android. Він має процесор, подібний до процесора ARM Cortex A15.

Цей SoC заснований на архітектурі ARM. Він має чотириядерний графічний процесор ARM Mali-400 MP4 1,4 ГГц та чотириядерний ARM Cortex - процесор A9. Цей процесор підтримує безліч програм, таких як 3D-ігри, багатозадачність та запис та відтворення відео.

SoC Medfield не засновані на архітектурі ARM. Він використовує технології x86 для створення цих SoC. Сокети Medfield можуть запропонувати OEM-виробникам одноядерний процесор 1,6-2 ГГц та графічний процесор SGX540 PowerVR.

Texas Instruments OMAP 4. Це четверте покоління OMAP, де використовується 45-нм архітектура ARM Cortex A9. Деякі пристрої Android, які використовують цей SoC, - це Motorola Atrix 2, Motorola Droid RAZR, LG Optimus 3D та LG Optimus Max [8].

2.4. Програмовані користувачем вентильні матриці – FPGA

FPGA це інтегральна схема, яка складається з внутрішніх апаратних блоків з програмованими користувачем межсоединения, щоб налаштувати роботу для конкретного додатка. Взаємозв'язки можна легко перепрограмувати, дозволяючи FPGA враховувати зміни в конструкції або навіть підтримувати нову програму протягом терміну служби деталі.

FPGA сягає своїм корінням у попередні пристрої, такі як програмовані пам’яті лише для читання (PROM) та програмовані логічні пристрої (PLD). Ці пристрої можна було запрограмувати як на заводі, так і в польових умовах, але вони використовували технологію запобіжників (отже, вираз «спалювання ПРОМ») і не могли бути змінені після програмування. На відміну від цього, FPGA зберігає інформацію про свою конфігурацію на перепрограмованому носії, такому як статична оперативна пам'ять (SRAM) або флеш-пам'ять.

Базова архітектура FPGA (рис. 2.10) складається з тисяч фундаментальних елементів, які називаються конфігурованими логічними блоками (CLB), оточеними системою програмованих взаємозв'язків, званою тканиною, яка направляє сигнали між CLB. Вхід / вихід (I / O) блокує інтерфейс між FPGA та зовнішніми пристроями [10,12].

Залежно від виробника, CLB може також називатися логічним блоком (LB),логічним елементом (LE) або логічною коміркою (LC).

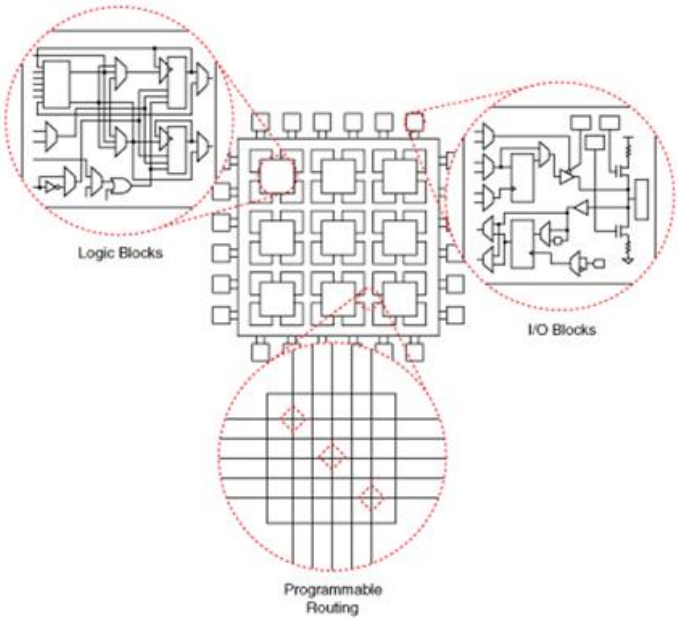


Рис. 2.10. Структура мікро- та наносхем сімейства FPGA

Окремий CLB (рис. 2.11) складається з декількох логічних блоків. Таблиця пошуку (LUT) є характерною особливістю FPGA. LUT зберігає заздалегідь визначений список логічних виходів для будь-якої комбінації входів: LUT з чотирма-шістьма вхідними бітами широко використовуються. Стандартні логічні функції, такі як мультиплексори (mux), повні суматори (FA) та тригери.

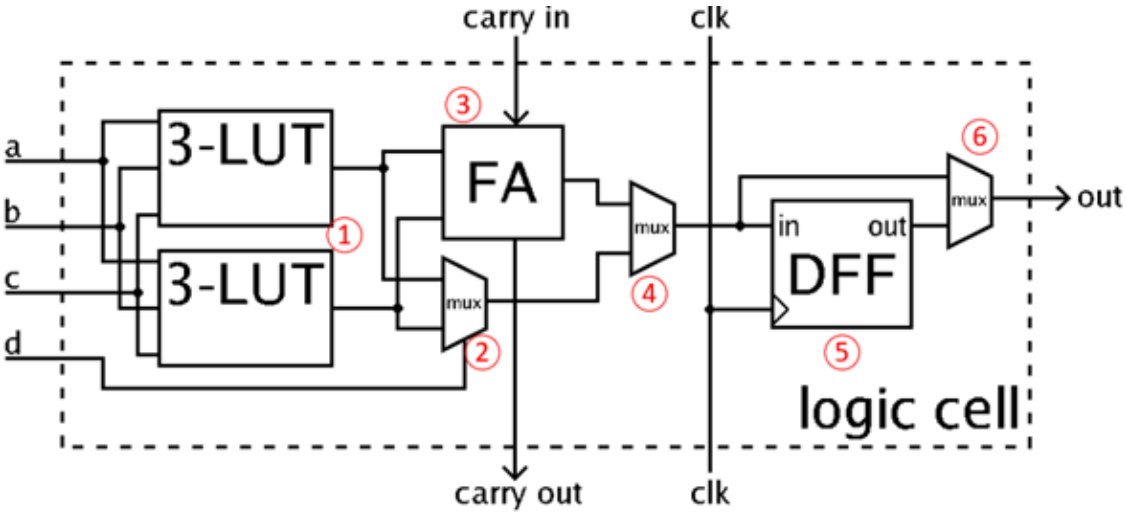


Рис. 2.11. Схема логічного елемента сімейства FPGA [10,12].

Сучасні програмовані логічні інтегральні схеми (ПЛІС, PLD - A programmable logic device, програмовані логічні пристрою - ПЛУ) - це вентильні матриці (ППВМ) або FPGA (field-programmable gate array). Вони містять інтерфейсні блоки, що конфігуруються логічні блоки CLB, і комутатори глобальних зв'язків – з інтерфейсними блоками: CBoxes - The connection box CBOX IN (CBOX OUT) і між CLB.

Локальні зв'язки реалізуються комутаторами всередині CLB - КЛБ.

Приклад КЛБ, конфигуруємого логічного блоку (CLB) представлений на рис. 2.12.

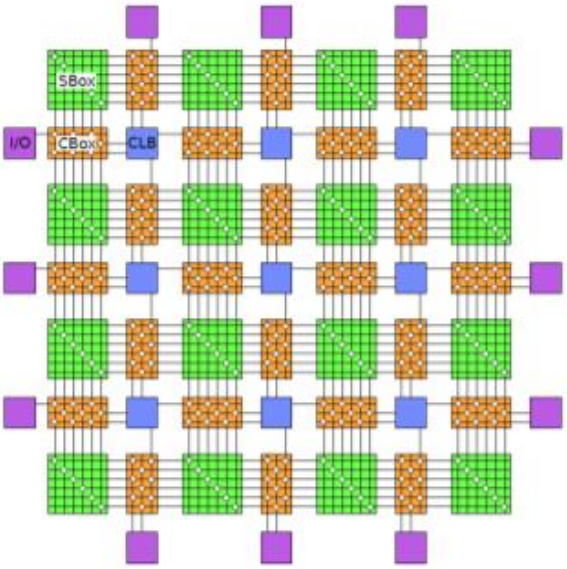


Рис.2.12. Архітектура ПЛІС FPGA

Логічний елемент (ЛЕ) являє собою постійний запам'ятовуючий пристрій ПЗУ (LUT - Look Up Table), виконаний на мультиплексорі, входи даних якого настроюються так званими конфігуруються осередками пам'яті.

Зазвичай LUT має 4 входи, LUT для n = 2 показаний рис.2.13.

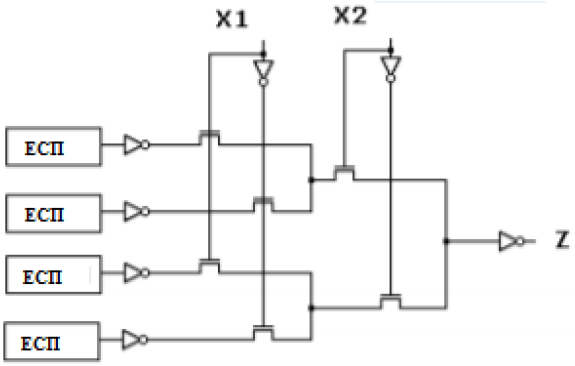


Рис.2.13. LUT для n = 2

Елементи статичної пам'яті SRAM - це так звана конфігураційна пам'ять, куди в цьому випадку записуються константи - таблиця істинності необхідної логічної функції.

Комутація зв'язків змінних всередині КЛБ для 4-х LUT представлена на рис.2.14.

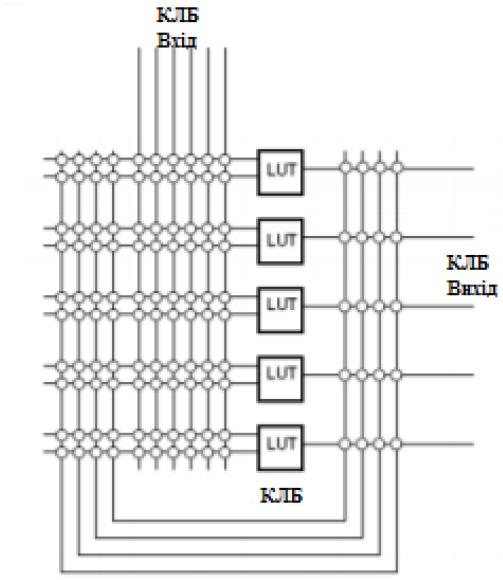


Рис.2.14. Комутація зв’язків 4-х LUT в середині КЛБ

ПЛІС надають широкі можливості реконфігурації логіки, проте все ще не в повній мірі використовується як для енергозбереження, так і для адаптації до відмов і відновлення логіки.

Енергозбереження в FPGA

Підходи до енергозбереження в FPGA багато в чому повторюють вище розглянуті загальні підходи, наприклад, характерні для процесорів, пам'яті та ін., включаючи використання згадуваного вище 14-нанометрового технологічного процесу Tri-Gate компанії Intel.

Особливістю FPGA є великі енерговитрати на комутатори (routing). За даними склад енергоспоживання 90-нанометровій ПЛІС Xilinx Spartan-3 має вигляд - рис.2.15.

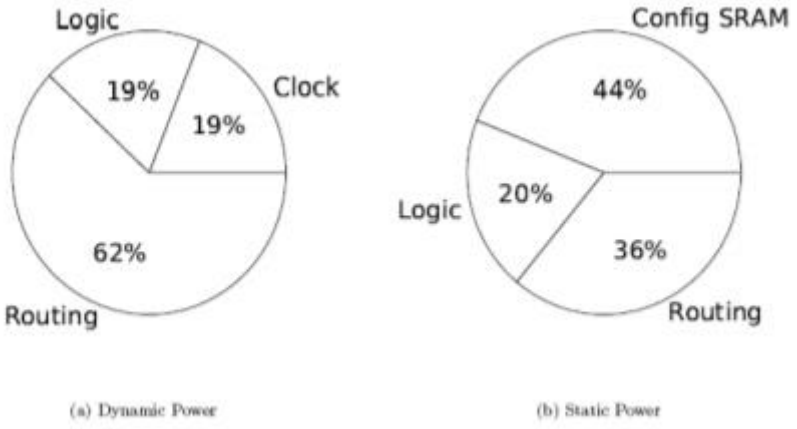


Рис.2.15. Склад енергоспоживання 90-нанометрової ПЛІС Xilinx Spartan-3

Блоки вводу-виводу FPGA**.** БВВ, зображений на прикладі мікросхеми сімейства Spartan фірми Xilinx (рис.2.16), забезпечує інтерфейс між виводами корпуса FPGA і її внутрішніми логічними схемами. Кожному виводу корпуса додається БВВ, який може бути конфігурований як вхід, вихід або двонаправлене з'єднання.

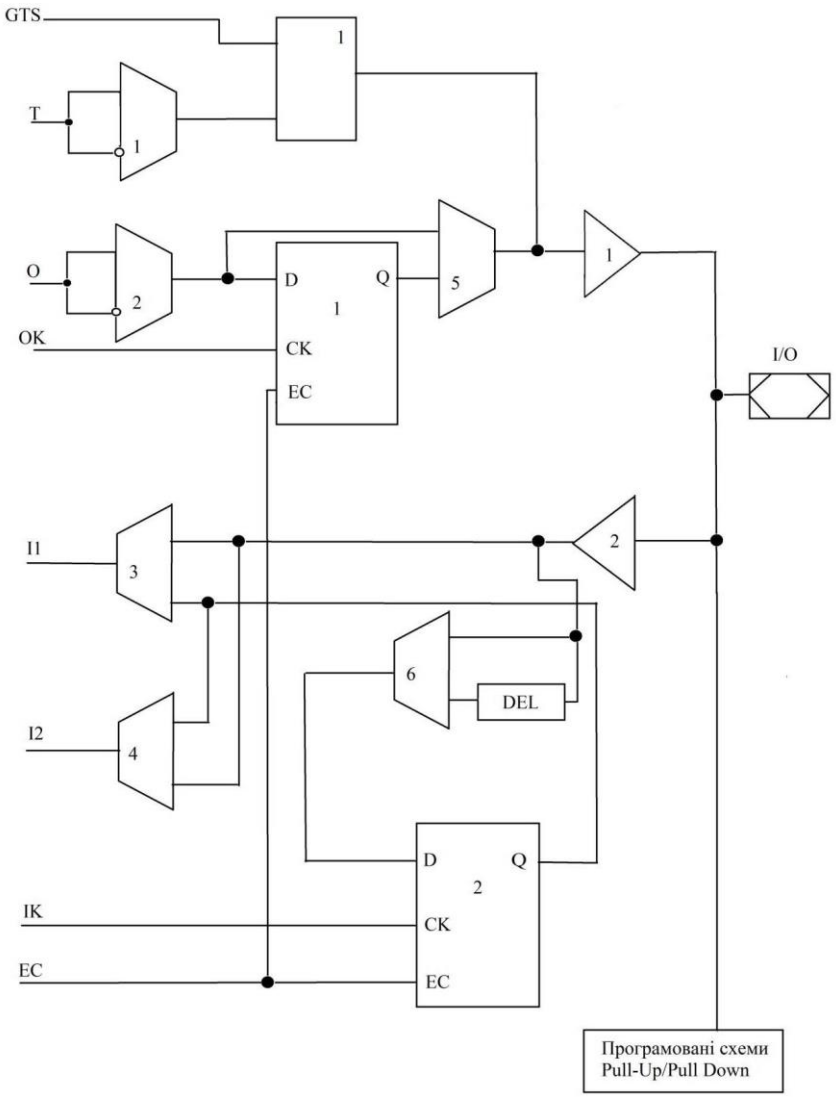
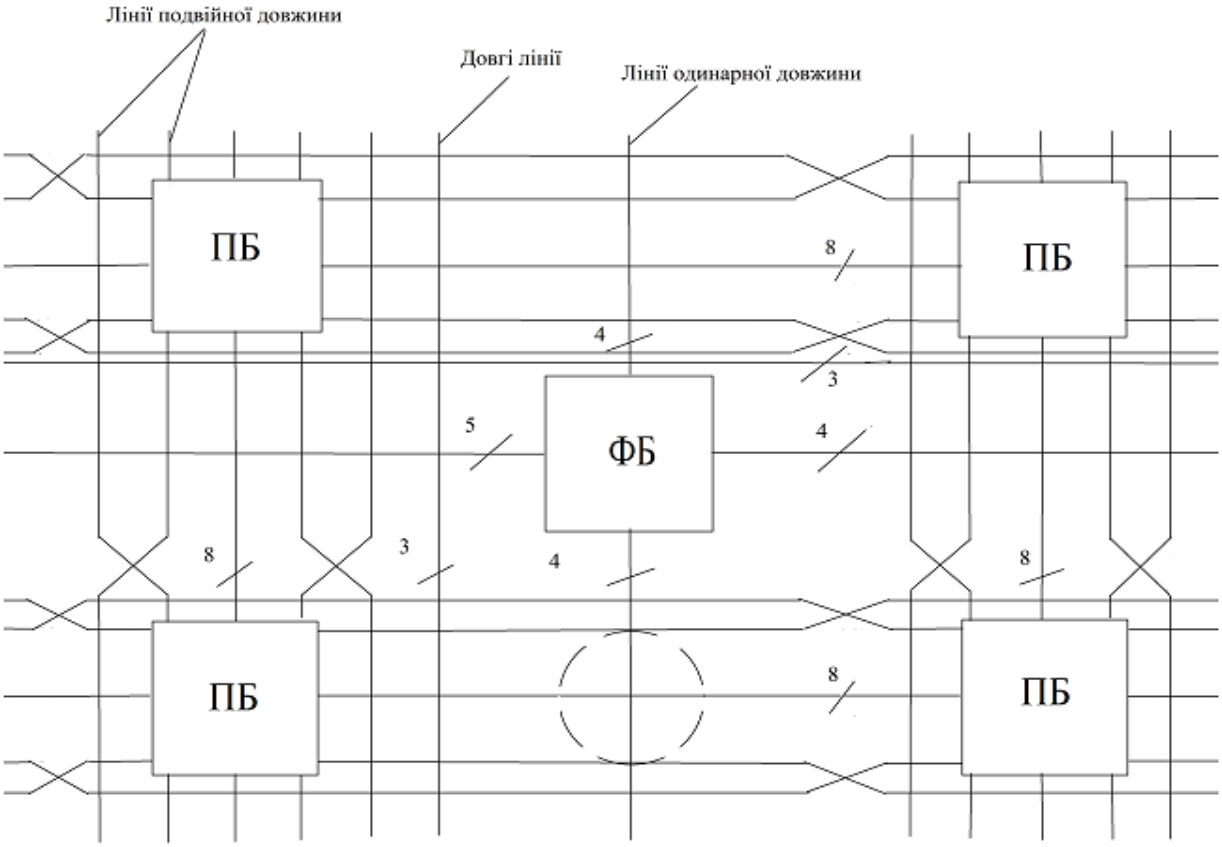
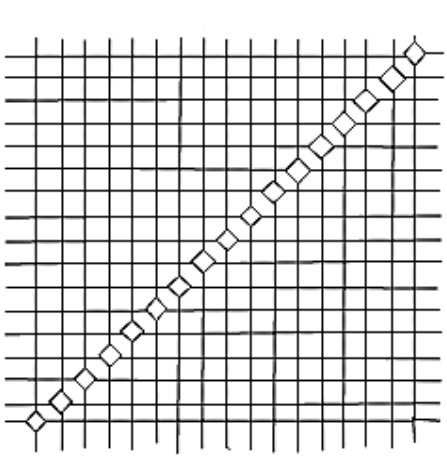


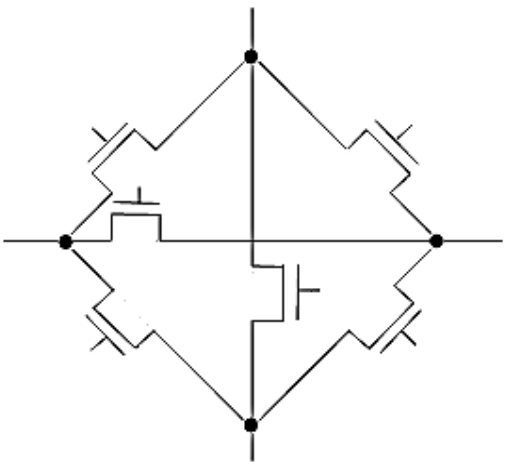
Рис. 2.16 – Приклад схеми блока вводу-виводу FPGA

Перемикаючі блоки ПБ та сегментовані лінії утворюють систему міжблочних з’єднань FPGA. ФБ мають квадратні геометричні обриси, їх виводи розподілені по всім сторонам квадрата для полегшення комутації. Для міжблочних з’єднань функціональних блоків у внутрішній області кристала є три типа зв’язку: одинарної довжини, подвійної довжини і довгі лінії. Спрощена система комутації FPGA показана на рис. 2.17,а.



а)





б) в)

Рис 2.17– Система комутаціїFPGA (а), схема перемикаючого блока (б) і схема длявстановленняз’єднанькомутуючихліній (в).

На перетинах вертикальних і горизонтальних каналів розташовані перемикаючі блоки (рис. 2.17, б). В межах ПБ перетинаються вертикальні та горизонтальні лінії зв'язку, і в кожному з перетинів є коло з 6 транзисторів для встановлення того чи іншого з'єднання (рис. 2.17, в). Сигнал, який надходить в ПБ по будь-якій лінії (наприклад, горизонтальній), може бути направлений вгору, вниз або прямо в залежності від того, який транзистор буде відкритий при конфігуруванні FPGA. Можлива і одночасна передача сигналу по декільком напрямкам, якщо необхідне його розгалуження.

До FPGA по прийнятій класифікації потрапляють мікросхеми з числом еквівалентних елементів біля 50 тис.-1 млн., системними частотами приблизно (50-80) МГц, числом користувацьких виходів 100-300. Лідером у виготовленні одноразово програмованих FPGA вважають фірму Actel, а у виготовленні FPGA з тригерною пам'яттю - фірму Xilinx.

2.5. Висновок до другого розділу

Чотири типи пристроїв класифікуються як ПЛІС. Сюди належать програмовані на місцях затворні масиви (FPGA), складні програмовані логічні пристрої (CPLD) та польові програмовані масиви затворних систем (на чіпі) (SoC), ПЛІС з комбінованою архітектурою (FLEX).

Хоча CPLD і FPGA також класифікуються як ПЛІС, вони також є стандартними напівпровідниковими інтегральними схемами (IC) - або мікросхемами. CPLD придатні для невеликих, нескладних, логічних завдань - на відміну від FPGA. CPLD та FPGA також відрізняються своєю внутрішньою архітектурою щодо того, як вони виконують свої логічні операції. У CPLD використовуються пошукові таблиці (LUT).

Новий підхід Intel до проектування FPGA відкрив кращу гнучкість дизайну, менші витрати на специфікацію матеріалів, менший розмір друкованих плат (друкованих плат), кращу надійність системи та більшу безпеку дизайну.

FPGA розроблені для високого рівня складності та широкого спектру можливостей інтеграції, а також доступні для стандартних конфігурацій та номерів

деталей. ПЛІС пропонують менші початкові витрати, швидший час виходу на ринок та підвищену гнучкість конструкції. Програмовані рішення забезпечують цінність для споживачів завдяки меншим витратам та енергоспоживанню.

Переваги ПЛІС:

- Програмовані логічні пристрої порівняно менші за розміром і займають менше місця на платі. Як результат, вони також мають менший час складання та відносно простіший процес складання сам по собі. Це також призводить до зменшення витрат.
- Програмовані логічні пристрої, як правило, споживають менший ступінь потужності, а також характеризуються тим, що мають менше наборів взаємозв'язків у пакетах порівняно з альтернативними варіантами. Усі ці функції перетворюються на велику надійність системи, а також гнучкість.
- PLD дуже програмовані на місцях - це означає, що мікросхему або схему можна запрограмувати повністю поза виробничим середовищем. Не обов'язково вносити зміни під час побудови схеми, оскільки модифікації можуть бути внесені пізніше, за потреби.
- Можливість змінювати конфігурацію пристрою робить програмовані логічні пристрої надзвичайно привабливим варіантом завдяки своїй настроюваності та персоналізації.
- Зменшена кількість інтегральних схем, що використовуються при заміні на PLD, підвищує надійність схеми, тим більше, що існує менша кількість міжмереж.
- Оскільки програмовані логічні пристрої стираються та змінюються, вони ідеально підходять для ситуацій або технологій, які потребують постійних оновлень, або якщо їх потрібно використовувати повторно в якийсь момент свого життєвого циклу.

Недоліки ПЛІС:

Обмеження архітектури для реалізації міжпроцесорних комунікацій. Основний проектний потік тільки дозволяє реалізувати загальну шину, та з'єднання типу «крапкакрапка». Та складні MPSoC можуть вимагати більш високої пропускної здатності ніж може запропонувати шина, або можуть вимагати більш ефективного використання ресурсів мікросхеми ніж за з'єднання «крапкакрапка» [8].

Відсутність ефективних механізмів сумісного застосування ресурсів. Як було вже зазначено засоби синхронізації від провідних виробників ПЛІС виявляються неефективним як метод синхронізації для високопродективних мультипроцесорних систем з великою кількістю процесорних ядер [7,8].

Обмежена кількість процесорних ядер. Кількість ядер, що можуть бути убудовані в один дизайн обмежені виробниками бібліотек, зокрема відсутністю в них ефективних механізмів синхронізації та когерентності для великої кількості процесорних ядер. Дослідники намагаються вирішити ці обмеження шляхом створення ad-hoc IP блоків та різноманітних функціональних модулів, що діють як доповнювання для вирішення певних проблем та обмежень інструментів EDK [8].

ПЛІС дозволили багатьом користувачам, дизайнерам та виробникам запропонувати неймовірно інноваційну та феноменальну технологію, яка зосереджена на створенні рішень, заснованих на логіці, у різних додатках. Знижене споживання енергії, менша вартість і інтеграція такої кількості функцій, які просто неможливі з більшістю інших альтернатив, роблять ПЛІС набагато улюбленішим та бажаним варіантом для кількох користувачів, що належать до ряду різних фонів і галузей.

РОЗДІЛ 3
АВТОМАТИЗОВАНЕ ПРОГРАМОВАННЯМУЛЬТИПЛЕКСОРНИХ МАЖОРИТАРНИХ НАНОСХЕМ

3.1. Програмування мультиплексорів в якості логічних елементів

Логічні елементи

Теоретичною основою цифрової техніки є алгебра логіки. Основним предметом булевої алгебри є просте твердження, яке або істинне (позначають символом 1) або хибне (позначають символом 0).

Прості висловлювання позначають буквами, які у цифровій техніці називають змінними (аргументами). Змінну зі скінченним числом значень називають перемикальною, а з двома значеннями – булевою.

Функція, яка має скінченне число значень називається перемикальною (логічною). Логічна функція, число можливих значень якої дорівнює двом, називається булевою. Булеві функції двох змінних подано в табл. 3.1.

Таблиця 3.1.

Двоаргументні логічні функції

ПОЗНАЧЕННЯ	НАБІР АРГУМЕНТІВ				НАЗВА ЛОГІЧНОЇ ФУНКЦІЇ	ФУНКЦІЯ
	0 0	0 1	1 0	1 1		
f_1	0	0	0	0	Константа нуль	0
f_2	0	0	0	1	Кон'юнкція	X_1X_2
f_3	0	0	1	0	Заборона X_2	$X_1\overline{X}_2$
f_4	0	0	1	1	Повторення X_1	X_1
f_5	0	1	0	0	Заборона X_1	$\overline{X}_1X_2$
f_6	0	1	0	1	Повторення X_2	X_2
f_7	0	1	1	0	Виключне АБО	$X_1 \oplus X_2 = X_1\overline{X}_2 \vee \overline{X}_1X_2$
f_8	0	1	1	1	Диз'юнкція	$X_1 \vee X_2$
f_9	1	0	0	0	Стрілка Пірса	$\overline{X_1 \vee X_2}$
f_{10}	1	0	0	1	Рівнозначність	$X_1 \sim X_2 = X_1X_2 \vee \overline{X}_1\overline{X}_2$
f_{11}	1	0	1	0	Інверсія X_2	$\overline{X}_2$
f_{12}	1	0	1	1	Імплікація від X_2 до X_1	$X_1 \vee \overline{X}_2$
f_{13}	1	1	0	0	Інверсія X_1	$\overline{X}_1$
f_{14}	1	1	0	1	Імплікація від X_1 до X_2	$\overline{X}_1 \vee X_2$
f_{15}	1	1	1	0	Штрих Шеффера	$\overline{X_1X_2}$
f_{16}	1	1	1	1	Константа одиниця	1

Фундаментальними логічними схемами є логічні елементи, з яких складаються складні схеми і цифрові системи.

Таблиця істинності – це спосіб подання залежності вихідних сигналів логічної схеми від логічних рівнів її вхідних сигналів. Таблиці істинності двозначних логічних функцій наведені у табл. 3.1.

Програмування мультиплексорів

Мультиплексори можливо використовувати в якості простих ПЛІС. У складніших випадках застосовують спеціально розроблені для цієї мети ПЛІС.

Найбільш просту реалізацію цієї ідеї вдається отримати з використанням мультиплексора, який окрім пристрою комутації може використовуватися і як універсальний логічний елемент. Суть використання мультиплексора в якості універсального логічного елементу полягає в тому, що його адресні входи використовуються як інформаційні і на них подаються аргументи відтворюваної функції, а інформаційні входи виконують роль програмованих (настроювальних).

При цьому на програмованих входах можуть формуватися сигнали або логічних констант, або деякі допоміжні функції.

Наприклад, реалізуємо на мультиплексорі (4→1) логічну функцію двох аргументів

Виключне АБО (додавання по модулю 2 чи нерівнозначності):

$$f_{\oplus}=x_1\bar{x}_0 \vee \bar{x}_1 x_0 = x_1 \oplus x_0, \qquad (3.1)$$

задану у вигляді таблиці істинності (табл. 3.2).

Таблиця 3.2

Таблиця істинності операції Виключне АБО

x_1	x_0	$f_{\oplus}$
0	0	0
0	1	1
1	0	1
1	1	0

Використовуючи рівняння вихідної функції цього мультиплексора:

$$f = D_3(x_1x_0) \vee D_2(x_1\bar{x}_0) \vee D_1(\bar{x}_1x_0) \vee D_0(\bar{x}_1\bar{x}_0), \quad (3.2)$$

його таблицю істинності (табл. 3.3):

Таблиця 3.3

Таблиця істинності мультиплексора (4→1)

x_1	x_0	f
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Та порівнюючи між собою функції (3.1), (3.2) і таблиці 3.2, 3.3, запишемо функцію програмування мультиплексора у вигляді:

$$f_{пр} = 0(x_1x_0) \vee 1(x_1\bar{x}_0) \vee 1(\bar{x}_1x_0) \vee 0(\bar{x}_1\bar{x}_0). \quad (3.3)$$

В результаті, реалізуємо цей алгоритм програмування у вигляді таблиці 3.4 і схеми мультиплексора, побудованої на рис. 3.1. Напруга живлення $U_{ж}$ реалізує лог.1, а заземлення –лог.0.

Таблиця 3.4

Таблиця програмування логічної функції Виключне АБО

x_1	x_0	$f_{пр}$	D	F
0	0	0	$D_0 = 0$	0
0	1	1	$D_1 = 1$	1
1	0	1	$D_2 = 1$	1
1	1	0	$D_3 = 0$	0

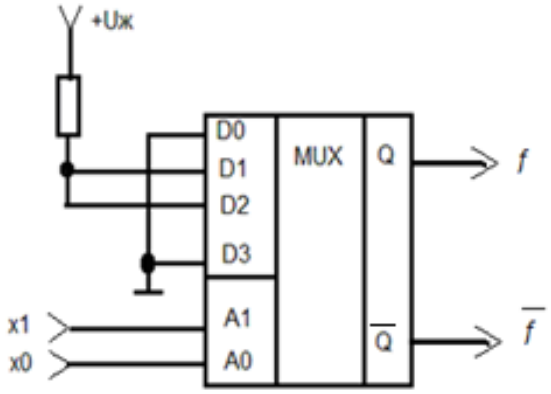


Рис. 3.1 - Реалізація логічної функції двох вхідних змінних на мультиплексорі

Отже, на мультиплексорі (4→1) можливе програмування 16 найпростіших функцій двох аргументів (табл. 3.4). Для цього на адресні входи A_1 та A_0 слід подавати бінарні аргументи x_1 та x_0, а інформаційні $D_3,\ldots,D_0$ - програмувати логічними константами цих аргументів з таблиць істинності.

З розглянутого прикладу видно, що описаний метод обмежується реалізацією функцій чотирьох змінних, оскільки мультиплексори, що реально випускаються, мають не більше 16 інформаційних входів. Таким чином, для реалізації $2^{2^4} = 2^{16} = 65536$ логічних функцій чотирьох вхідних змінних можна скористатися ІС мультиплексора, яка буде універсальним логічним елементом, оскільки без використання додаткових апаратних засобів дозволяє реалізувати логічну функцію довільного виду.

Для порівняння приклад реалізації операції (3.1) з використанням п’яти елементів НІ, І та АБО наведений на рис. 3.2.

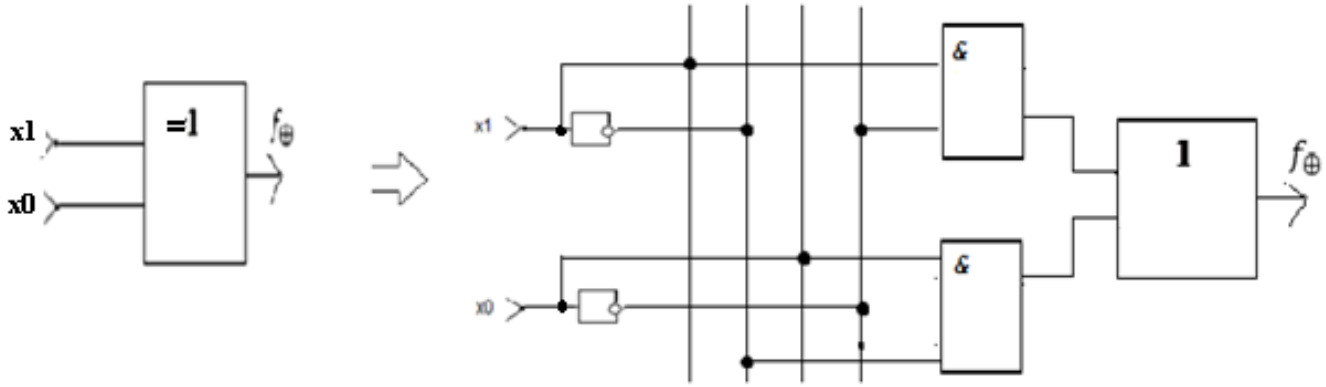


Рис.3.2 – Мікросхемна реалізація операції Виключне АБО

При необхідності реалізації логічної функції великого числа вхідних змінних можна скористатися структурою мультиплексорного дерева. Проте при невеликому числі аргументів цю задачу можна вирішувати і іншим методом, а саме вибором сигналів налаштування не з множини{1,0}, як це було зроблено вище, а з множини $\{1,0,\ x_i\}$, де x_i - один з аргументів відтворюваної функції. В цьому випадку іноді вдається на мультиплексорі без додаткових апаратних витрат реалізувати логічну функцію, число аргументів якої на одиницю більше числа його адресних входів.

Наприклад, на мультиплексорі (4→1) реалізуємо логічну функцію трьох вхідних змінних наступного виду:

$$f = x_2 \vee x_1\bar{x}_0 \vee \bar{x}_1x_0\ . \qquad (3.4)$$

Для отримання диз'юнктивної нормальної форми (ДНФ) логічних функцій використовують дужкові перетворення, тобто неповні добутки (терми) домножують на одиночні суми недостаючих аргументів та функцій: $(x_i \vee \bar{x}_i)$, $(x_i \vee\ 1)$, $(f_i \vee \bar{f}_i)$, $(f_i \vee 1)$ або додають нульові добутки $(x_i\ \bar{x}_i)$, $(x_i\ 0)$, $(f_i\ \bar{f}_i)$, $(f_i\ 0)$.

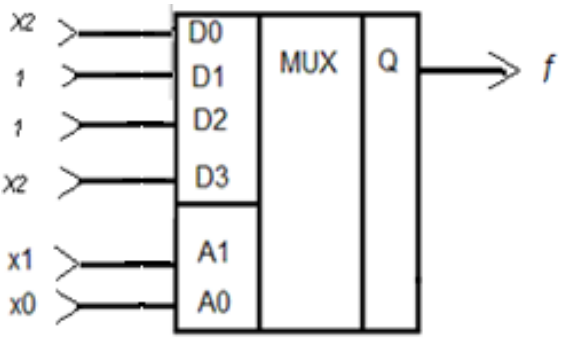
Перетворимо за цими правилами задану функцію (3.4):

$$f = x_2(x_1 \vee \bar{x}_1)(x_0 \vee \bar{x}_0) \vee x_1\bar{x}_0 \vee \bar{x}_1x_0 = x_2x_1x_0 \vee x_2x_1\bar{x}_0 \vee$$
$$x_2\bar{x}_1x_0 \vee x_2\bar{x}_1\bar{x}_0 \vee x_1\bar{x}_0 \vee \bar{x}_1x_0 = x_2x_1x_0 \vee x_1\bar{x}_0(x_2 \vee 1) \vee \bar{x}_1x_0(x_2 \vee 1) \vee \qquad (3.5)$$
$$x_2\bar{x}_1\bar{x}_0 = x_2(x_1x_0) \vee 1(x_1\bar{x}_0) \vee 1(\bar{x}_1x_0) \vee x_2(\bar{x}_1\bar{x}_0).$$

Отже, отримано функцію програмування (3.5) мультиплексора (4→1), у якого на адресні входи подано сигнали змінних x_1 і x_0. Таблиця і схема програмування у цьому прикладі наведені на рис. 3.3

x_1	x_0	D
0	0	$D_0 = x_2$
0	1	$D_1 = 1$
1	0	$D_2 = 1$
1	1	$D_3 = x_2$

а)



б)

Рис. 3.3 – Таблиця програмування (а) і схемна реалізація (б) функції трьох вхідних змінних

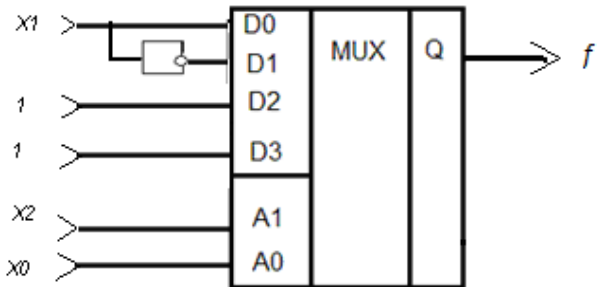
Слід зауважити, що таке схемотехнічне рішення не є єдиним. На адресні входи A_1 та A_0 можна подавати довільні аргументи, а на програмовані входи $D_3,\ldots,D_0$ - залишкові аргументи чи навіть функції.

Реалізуємо початкову логічну функцію (3.4), використовуючи в якості адресних сигналів x_2 і x_0. Зробимо дужкові перетворення з урахуванням (3.1) і отримаємо функцію програмування мультиплексора:

$$f_{пр} = x_2(x_0 \vee \bar{x}_0) \vee f_{\oplus}(x_2 \vee \bar{x}_2) = x_2x_0 \vee x_2\bar{x}_0 \vee x_2x_1\bar{x}_0 \vee x_2\bar{x}_1x_0 \vee$$
$$\bar{x}_2x_1\bar{x}_0 \vee \bar{x}_2\bar{x}_1x_0 = (1 \vee \bar{x}_1)x_2x_0 \vee (1 \vee \bar{x}_1)x_2\bar{x}_0 \vee \bar{x}_1(\bar{x}_2x_0) \vee x_1(\bar{x}_2\bar{x}_0) = \quad (4.6)$$
$$1(x_2x_0) \vee 1(x_2\bar{x}_0) \vee \bar{x}_1(\bar{x}_2x_0) \vee x_1(\bar{x}_2\bar{x}_0).$$

Таблиця і схема програмування другого варіанту показані на рис. 3.4.

x_2	x_0	D
0	0	$D_0 = x_1$
0	1	$D_1 = \bar{x}_1$
1	0	$D_2 = 1$
1	1	$D_3 = 1$



а) б)

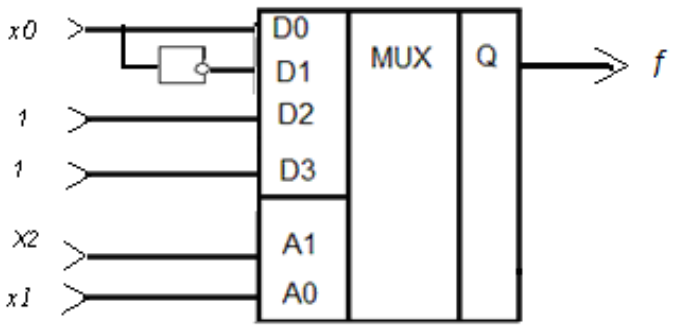
Рис. 3.4 – Таблиця програмування (а) і схемна реалізація (б) для другого варіанта програмування функції трьох вхідних змінних

Тепер втретє запрограмуємо мультиплексор для виконання заданої функції (3.4) при використанні в якості адресних аргументів x_2 і x_1. Виконаємо дужкові перетворення:

$$f = x_2 \vee x_1\bar{x}_0 \vee \bar{x}_1x_0 = x_2(x_1 \vee \bar{x}_1) \vee f_{\oplus}(x_2 \vee \bar{x}_2) = x_2x_1 \vee x_2\bar{x}_1 \vee$$
$$x_2x_1\bar{x}_0 \vee x_2\bar{x}_1x_0 \vee \vee\bar{x}_2x_1\bar{x}_0 \vee \bar{x}_2\bar{x}_1x_0 = 1(x_2x_1) \vee 1(x_2\bar{x}_1) \vee \bar{x}_0(\bar{x}_2x_1) \vee \quad (3.7)$$
$$x_0(\bar{x}_2\bar{x}_1).$$

На рис. 3.5 наведені таблиця істинності (а) та схемотехнічна реалізація (б) третього варіанту програмування (3.7) функції трьох аргументів (3.4).

x_2	x_i	D
0	0	$D_0 = x_0$
0	1	$D_1 = \bar{x}_0$
1	0	$D_2 = 1$
1	1	$D_3 = 1$



а) б)

Рис. 3.5 – Таблиця програмування (а) і схемна реалізація (б) третього варіанта програмування функції трьох вхідних змінних

Аналіз перевірочної таблиці істинності (рис.3.5,а) свідчить, що при любій комбінації адресних аргументів на виході мультиплексора формується задана функція програмування (3.4).

Таблиця 4.5

Таблиця істинності трьох варіантів програмування функції трьох аргументів

x_2	$\bar{x}_2$	x_1	$\bar{x}_1$	x_0	$\bar{x}_0$	$f=(x_1x_0)$	$f=(x_2x_0)$	$f=(x_2x_1)$
0	1	0	1	0	1	0	0	0
0	1	0	1	1	0	1	1	1
0	1	1	0	0	1	1	1	1
0	1	1	0	1	0	0	0	0
1	0	0	1	0	1	1	1	1
1	0	0	1	1	0	1	1	1
1	0	1	0	0	1	1	1	1
1	0	1	0	1	0	1	1	1

Очевидно, що в двох останніх випадках (рис. 3.4,б та 3.5,б) практична реалізація заданої логічної функції (3.4) вимагає введення додаткового логічного елементу НІ. З точки зору апаратних витрат перше схемотехнічне рішення (рис. 3.3,б) є оптимальним. Отже, коли число адресних входів мультиплексора на один менше числа незалежних змінних заданої логічної функції, елемент не є універсальним, а стає багатофункціональним.

При розробці складних логічних пристроїв доводиться стикатися з виконанням однотипних операцій І та АБО над різною кількістю змінних. В якості прикладів запрограмуємо мультиплексори (4→1) на виконання логічних функцій помноження та додавання декількох аргументів:

$$f_I = x_n \dots x_2 x_1 x_0 = \bigwedge_{i=0}^{n} x_i, \qquad (3.8)$$

$$f_{АБО} = x_n \vee \ldots \vee x_2 \vee x_1 \vee x_0 = \bigvee_{i=0}^{n} x_i. \qquad (3.9)$$

Після дужкових перетворень функції (3.8) та після домножень суми аргументів $[x_n \vee \ldots \vee x_2\ ]$ на одиночні суми недостаючих аргументів $(x_1 \vee \bar{x}_1)$, $(x_0 \vee \bar{x}_0)$, аргументів x_1 - на $(x_0 \vee \bar{x}_0)$ та x_0 - на $(x_1 \vee \bar{x}_1)$ функції (3.9), отримують відповідні функції програмувань:

$$f_{пр\,I} = \left[\bigwedge_{i=2}^{n} x_i\right](x_1x_0) \vee 0(x_1\bar{x}_0) \vee 0(\bar{x}_1x_0) \vee 0(\bar{x}_1\bar{x}_0), \qquad (3.10)$$

$$f_{пр.АБО} = \left[\bigvee_{i=2}^{n} x_i\right](x_1 \vee \bar{x}_1)(x_0 \vee \bar{x}_0) \vee x_1(x_0 \vee \bar{x}_0) \vee x_0(x_1 \vee \bar{x}_1) =$$

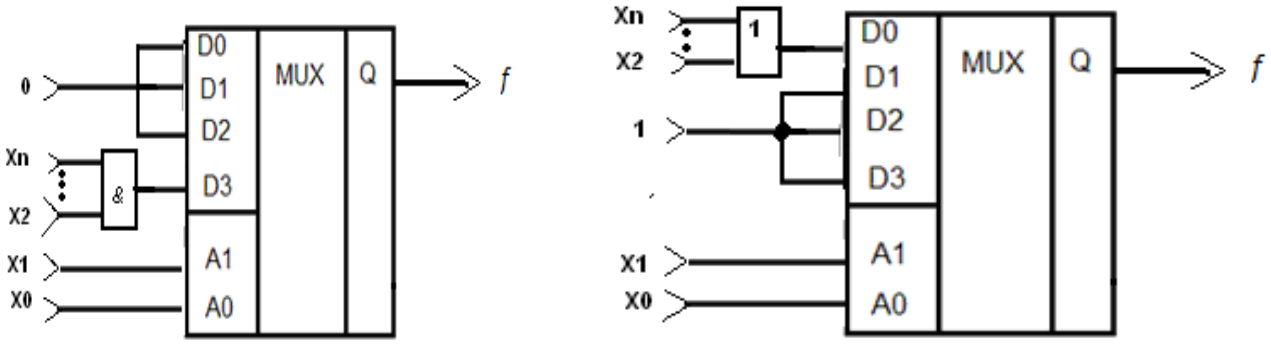
$$= \left[\bigvee_{i=2}^{n} x_i\right](x_1x_0 \vee x_1\bar{x}_0 \vee \bar{x}_1x_0 \vee \bar{x}_1\bar{x}_0) \vee (x_1x_0 \vee x_1\bar{x}_0 \vee x_1x_0 \vee \bar{x}_1x_0) =$$

$$= \left(\left[\bigvee_{i=2}^{n} x_i\right] \vee 1 \vee 1\right)x_1x_0 \vee \left(\left[\bigvee_{i=2}^{n} x_i\right] \vee 1\right)x_1\bar{x}_0 \vee \left(\left[\bigvee_{i=2}^{n} x_i\right] \vee 1\right)\bar{x}_1x_0 \vee \left[\bigvee_{i=2}^{n} x_i\right]\bar{x}_1\bar{x}_0 = \qquad (3.11)$$

$$=$$

$$= 1(x_1x_0) \vee 1(x_1\bar{x}_0) \vee 1(\bar{x}_1x_0) \vee \left[\bigvee_{i=2}^{n} x_i\right]\bar{x}_1\bar{x}_0.$$

На рис. 3.6 наведені схемотехнічні реалізації задач програмування функції (3.8) та (3.9) на мультиплексорах (4→1).



а) б)

Рис. 3.6 – Схемні реалізації програмування операцій І (а) та АБО (б)

Далі, з використанням отриманих результатів, реалізуємо на мультиплексорі (4→1) складнішу логічну функцію чотирьох аргументів:

$$f = x_3x_1\bar{x}_0 \vee x_2\bar{x}_1x_0 \vee x_3x_0. \qquad (3.12)$$

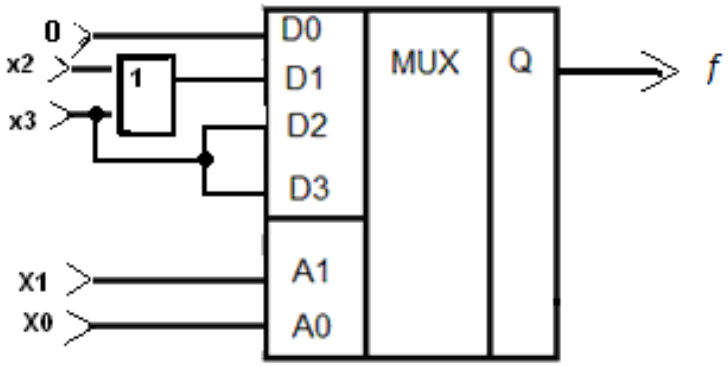
Оберемо в якості адресних змінних x_1 та x_0 і тому домножимо останній неповний добуток (терм) x_3x_0 на одиночну суму недостаючої змінної $(x_1 \vee \bar{x}_1)$:

$$f = x_3x_1\bar{x}_0 \vee x_2\bar{x}_1x_0 \vee x_3x_0(x_1 \vee \bar{x}_1) = \\ = x_3(x_1x_0) \vee x_3(x_1\bar{x}_0) \vee (x_3 \vee x_2)(\bar{x}_1x_0) \vee 0(\bar{x}_1\bar{x}_0). \quad (3.13)$$

За отриманою функцією (3.13) на рис. 3.7 побудовані таблиця (а) та схема (б) програмування мультиплексора, що реалізує функцію (3.12).

x_2	x_i	D
0	0	$D_0 = 0$
0	1	$D_1 = x_3 \vee x_2$
1	0	$D_2 = x_3$
1	1	$D_3 = x_3$

а)



б)

Рис. 3.7 – Таблиця програмування (а) і схемотечнічна реалізація (б) функції чотирьох вхідних змінних

За допомогою такого методу, можливо обирати сигнали програмування з ширшої множини, до якої входить декілька аргументів. При цьому на мультиплексорі з двома адресними входами можна ралізувати логічну функцію трьох, чотирьох і більше змінних. Ефективність використання такого технічного рішення зі збільшенням числа вхідних змінних падає.

3.2. Програмовані мультиплексорні наносхеми на базі мажоритарних елементів

Постановка задачі №1

1. Записати логічні функції на виходах ПНЕС, мажоритарному та булевому базисах.
2. Реалізувати комп'ютерне моделювання ПНЕС на САПР QCA Designer для виконання логічних функцій.
3. Розрахувати загальну кількість КА, їх розміри та відстані між їх центрами, діаметри

квантових острівців і загальні розміри ПНЕС.

4. Дослідити залежність функцій від температури.
5. Побудувати повну таблицю істинності для всіх проміжних та кінцевих виводів ПНЕС на КА.

Фізичні основи роботи наноелектронних квантових автоматів (КА)

Електростатична енергія, яка накопичена вструктурі КА:

$$E = \min\{\frac{4Q}{C_{\Sigma}} - nqU_{вх}\} \qquad (3.14)$$

буде мати мінімальне значення тільки тоді, коли два електрони (n=2) знаходяться в діагональних квантових точках. В рівнянні (3.14) Q та C_{Σ} - величина заряду та загальна ємність рівноцінних квантових точок.

Через кулонівське відштовхування, яке спостерігається при T = 2мК, два електрони в КА можуть знаходитись тільки у двох стабільних станах, які зображено на рис. 3.8.

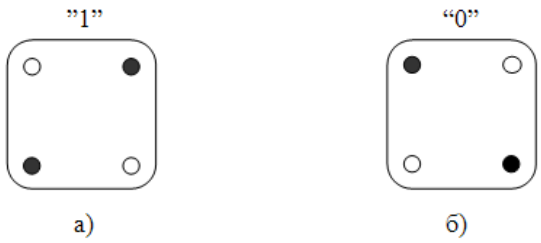
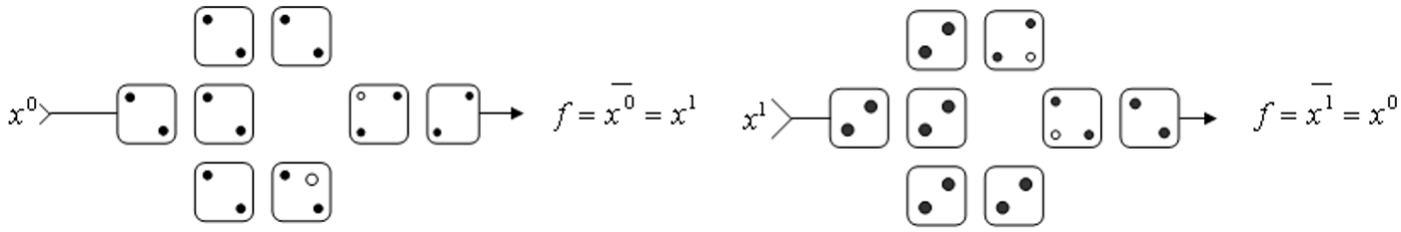
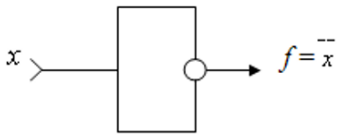


Рис. 3.8. Діагональне розміщення електронів на квантових точках в КА, які відповідають логічній одиниці (а) та нулю (б)

Для створення логічного інвертора на КА використовують чотири послідовно-паралельно розташовані лінії передачі (рис. 3.9, а).



а)



б)

Рис. 3.9. Наносхема виконання логічної операції заперечення НІ на КА (а) та умовне позначення інвертора (б)

Таке підключення КА забезпечує інверсію сигналу за рахунок кулонівського відштовхування електронів між кутовими квантовими точками автоматів у вихідних лініях схеми (рис. 3.9., а). Таким чином, по дві квантові точки залишаються незаповненими.

Як і практично в усій наноелектроніці, найбільше застосування КА знаходять у схемах мажоритарного вибору. На рис. 3.10. наведена повна електрична схема мажоритарного елемента на базі дев'яти КА.

В центральній комірці (рис. 3.10, а) під впливом електростатичної індукції, яку наводять оточуючі верхня, зліва та нижня сусідні комірки, формується діагональне розташування електронів, яке відповідає логічному нулю або одиниці за принципом вибору більшості. У подальшому цей логічний стан передається вправо через останню комірку на вихід схеми. Таким чином виконується логічна функція мажоритарного вибору (2 з 3):

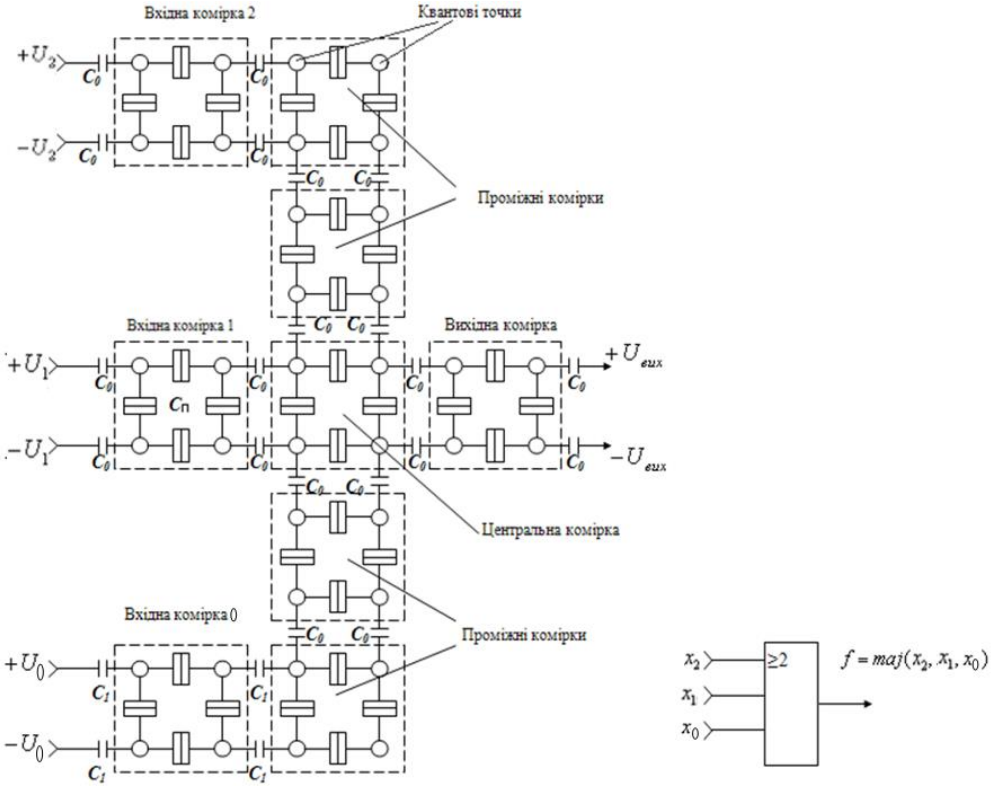
$$f = maj(x_2, x_1, x_0) = x_2x_1 \vee x_2x_0 \vee x_1x_0. \tag{3.15}$$

Якщо на третьому вході МЕ (рис. 3.9, б) постійно підтримувати логічний нуль чи одиницю, то схема в цілому здійснює логічні операції І чи АБО. В табл. 1 наведені стани входів й виходу МЕ, розподіл надлишкових електронів у комірках спрощених схем МЕ, та умовні позначення елементів.

Для створення універсального МЕ (УМЕ) з прямим та інверсним виходами, електричну схему з табл. 1 доповнюють схемою інвертора (рис. 3.10, а). Логічна функція інверсного виходу описується рівнянням:

$$\overline{f} = maj\ (\bar{x}_2, \bar{x}_1, \bar{x}_0) = \bar{x}_2\ \bar{x}_1 \vee\ \bar{x}_2\ \bar{x}_0 \vee\ \bar{x}_1\ \bar{x}_0\,. \tag{3.16}$$

Інколи функцію (3.16) називають мінорітарною $\bar{f} = min(x_2, x_1, x_0)$.



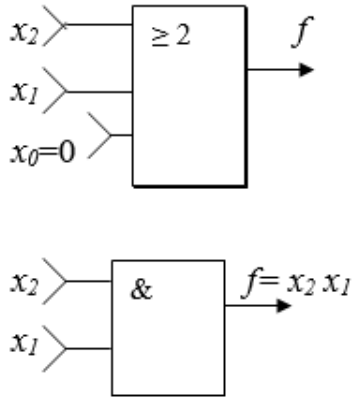
а) б)

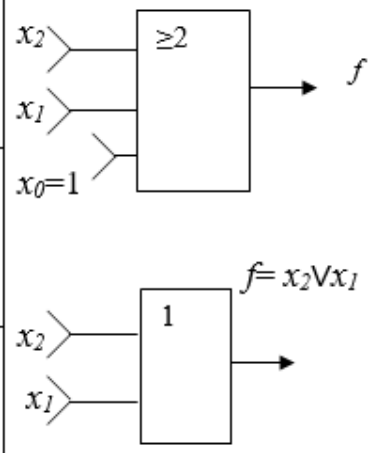
Рис. 3.10 Електрична схема мажоритарного наноелемента (2 з 3) на квантових точках (а) та його умовне позначення (б)

Таблиця 4.6

Побудова логічних схем І та АБО з використанням мажоритарних елементів на квантових автоматах

Таблиця дійсності				Розподіл електронів в КА	Умовне позначення схем
x_2	x_1	x_0	f		
0	0	0	0	x_2=0, x_1=0, x_0=0 → f=0	
0	1	0	0	x_2=0, x_1=1, x_0=0 → f=0	
1	0	0	0	x_2=1, x_1=0, x_0=0 → f=0	
1	1	0	1	x_2=1, x_1=1, x_0=0 → f=1	
0	0	1	0	x_2=0, x_1=0, x_0=1 → f=0	
0	1	1	1	x_2=0, x_1=1, x_0=1 → f=1	
1	0	1	1	x_2=1, x_1=0, x_0=1 → f=1	
1	1	1	1	x_2=1, x_1=1, x_0=1 → f=1	





На рис. 3.11 показані схеми УМЕ на КА (а) і також логічних елементів, які здійснюють операції 2І-НІ (б) та 2АБО-НІ (в).

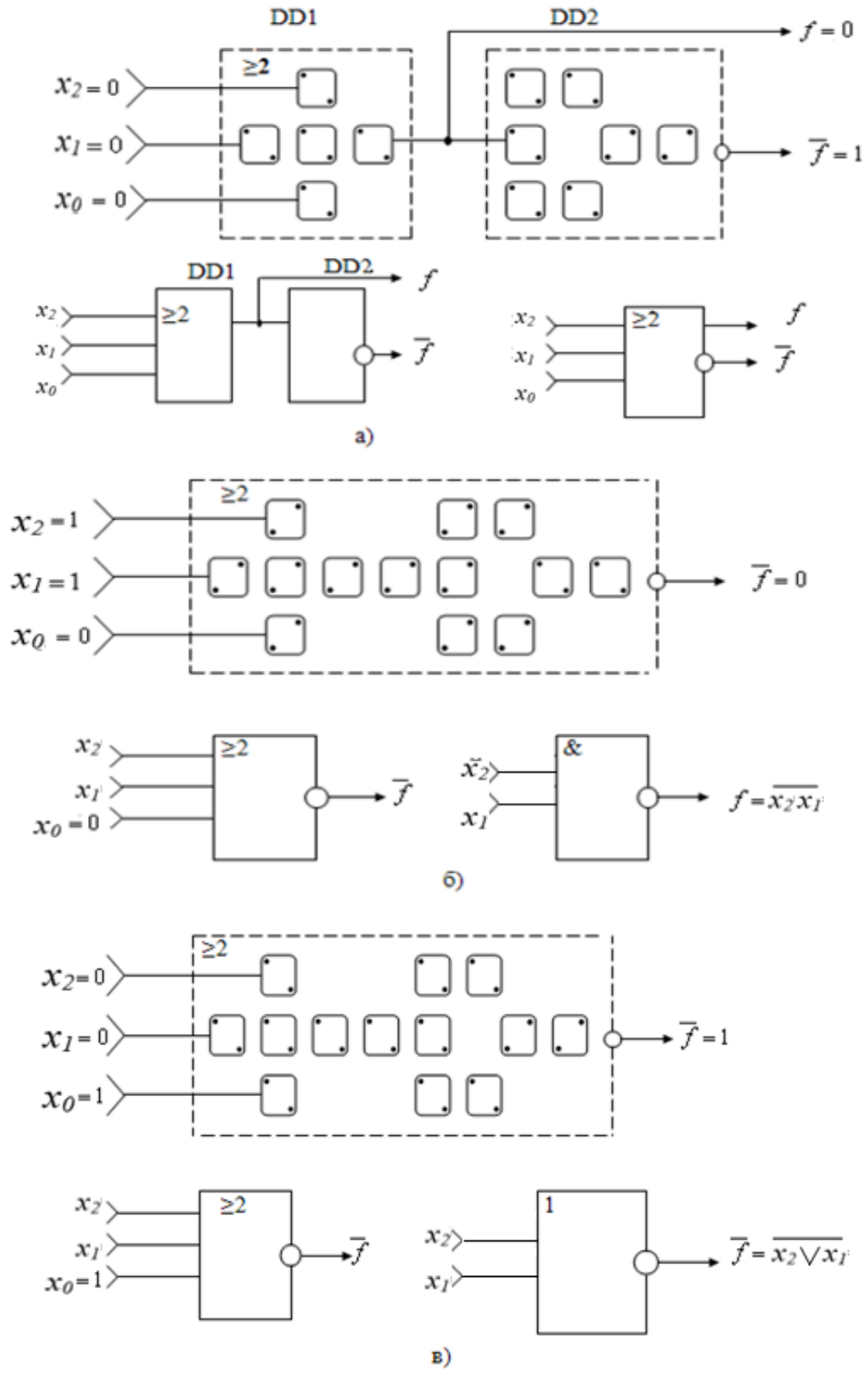


Рис.3.11

Вирішення поставленої задачі №1 кваліфікаційної роботи

1. Записуємо логічні функції на виходах ПНЕС, мажоритарному та булевому базисах.

Задано схему:

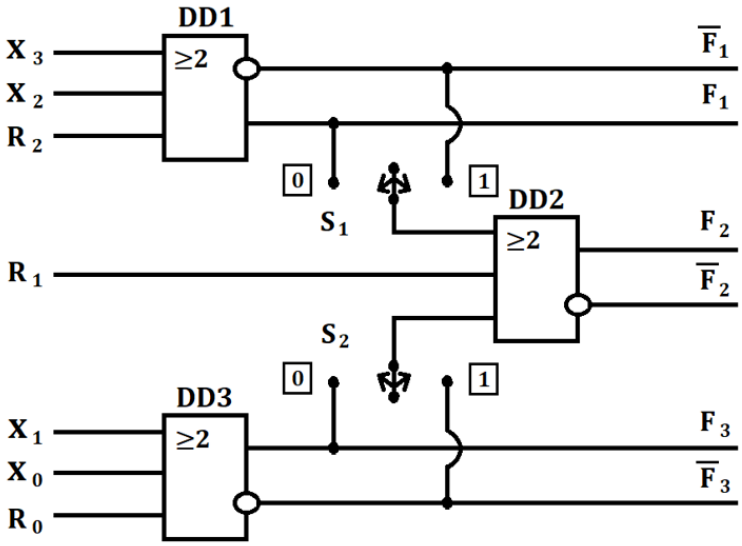


Рис.3.12

Перетворимо ключі на програмовані мультиплексори

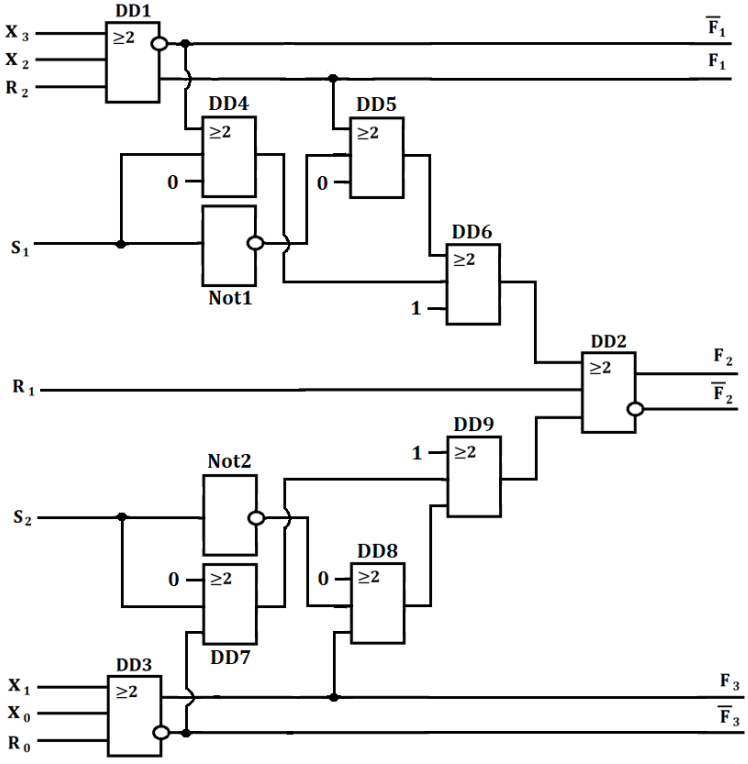


Рис.3.13

Рівняння для усіх логічних функцій ПНЕС в мажоритарному та булевому базисах:

$$f_1 = maj(x_3, x_2, 1) = x_3 \vee x_2, \tag{3.17}$$

$$\overline{f_1} = maj(\overline{x_3}, \overline{x_2}, 0) = \overline{x_3}\,\overline{x_2}, \tag{3.18}$$

$$f_2 = maj[maj(x_3, x_2, 1), maj(x_1, x_0, 0), 0] = (x_3 \vee x_2)(x_3 x_2)\,, \tag{3.19}$$

$$\overline{f_2} = maj[maj(\overline{x_3}, \overline{x_2}, 0), maj(\overline{x_1}, \overline{x_0}, 1), 1] = (\overline{x_3}\,\overline{x_2}) \vee (\overline{x_1} \vee \overline{x_0}), \tag{3.20}$$

$$f_3 = maj(x_1, x_0, 0) = x_1 x_0, \tag{3.21}$$

$$\overline{f_3} = maj(\overline{x_1}, \overline{x_0}, 1) = \overline{x_1} \vee \overline{x_0}, \tag{3.22}$$

2. Реалізовуємо комп'ютерне моделювання ПНЕС на САПР QCA Designer для виконання логічних функцій.

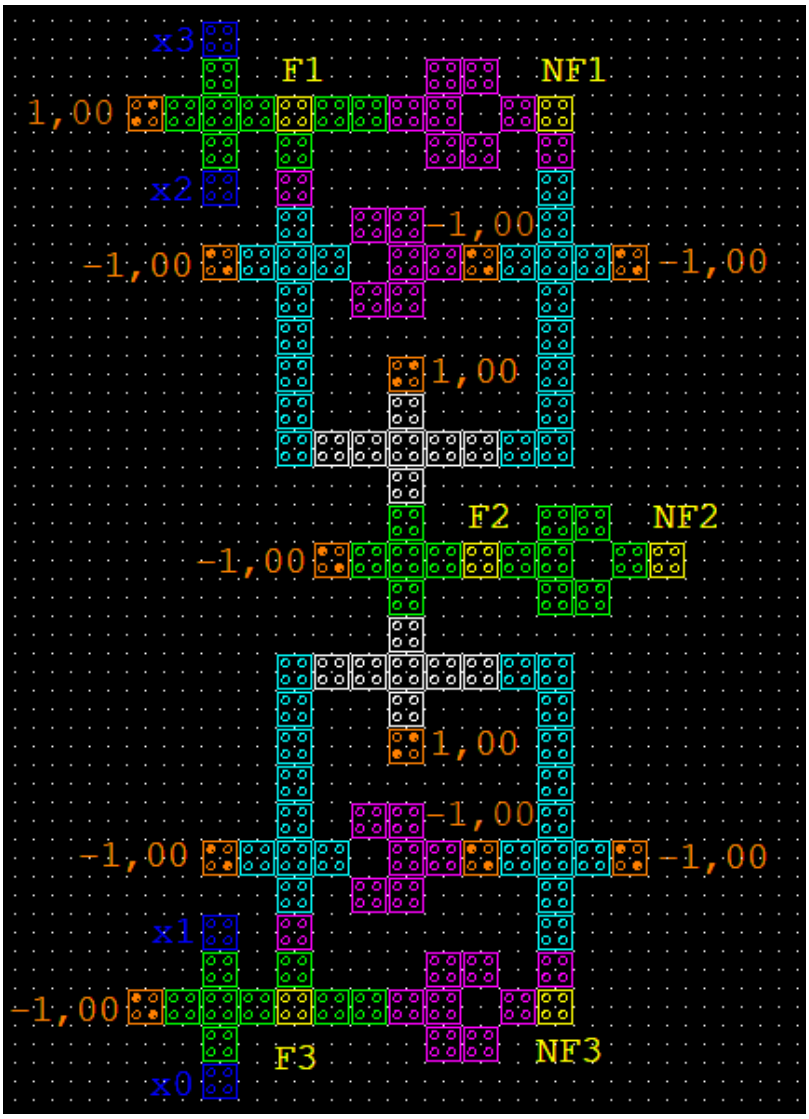


Рис.3.14. Моделювання ПНЕС на САПР QCA Designer

3. Розраховуємо загальну кількість КА, їх розміри та відстані між їх центрами, діаметри квантових острівців і загальні розміри ПНЕС.

   Виходи F_2 та $\bar{F}_2$ мають ссув на 180°.

- Загальна кількість квантових комірок: 133, у яких 4 входи та 6 виходи
- Розмір квантової комірки: (18x18)нм
- Відстань між центрами КА: 20нм

- Діаметр квантового острівця: 5нм

Загальні розміри наносхеми: (412x587) нм = 241844 нм^2 = 0,24 мкм^2

4. Досліджуємо залежність функцій від температури

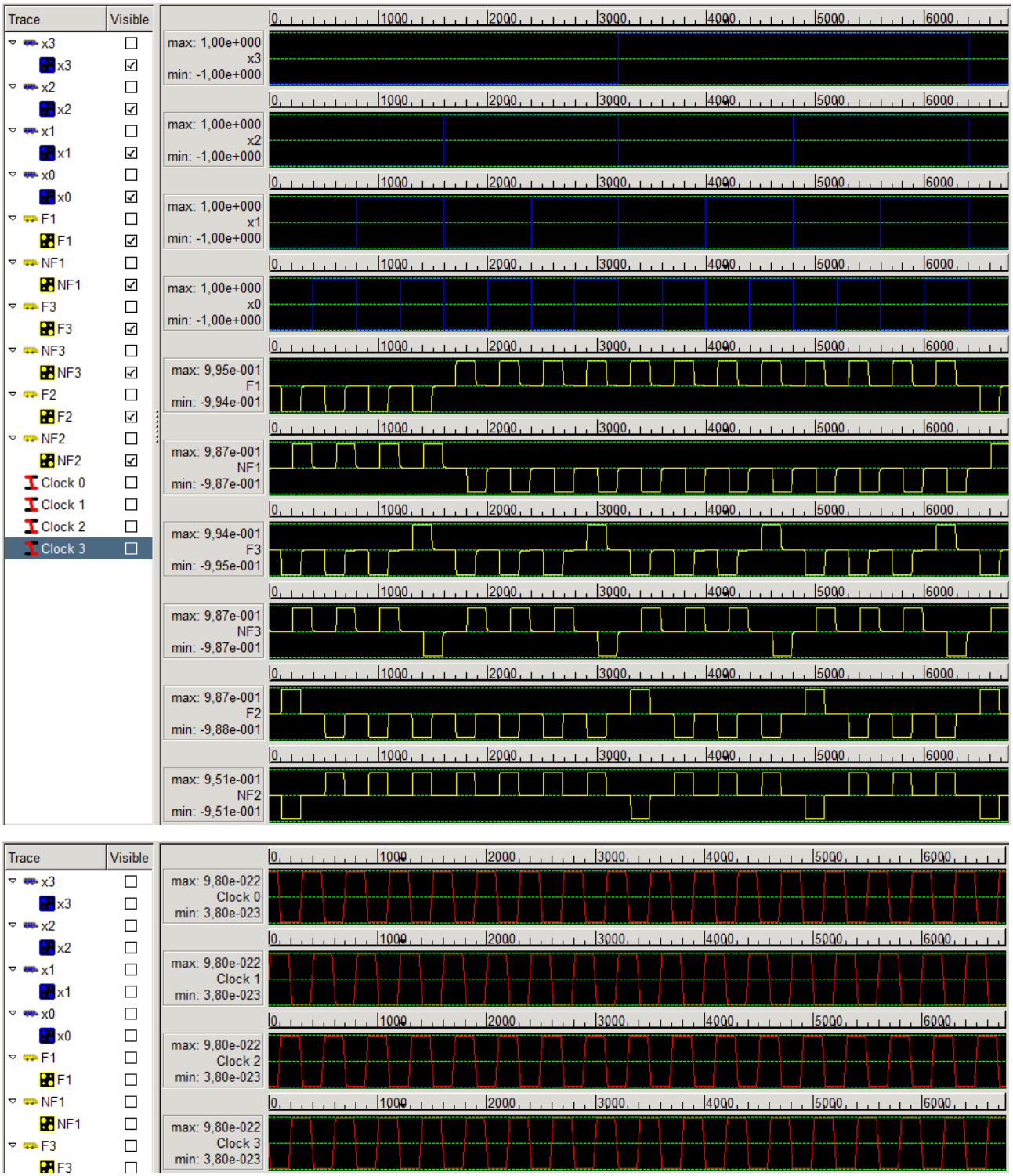


Рис.3.15. Часовий аналіз запрограмованої наносхеми

Оскільки, функції F1, F2, F3, та NF1, NF2, NF3 під час температурного дослідження змінювалися однаково, то відобразимо зміни на прикладі функцій F1 та NF1.

Стан функцій:

при температурі T=0-10 К (рис.3.16)

Рис.3.16

при T=20К (рис.3.17)

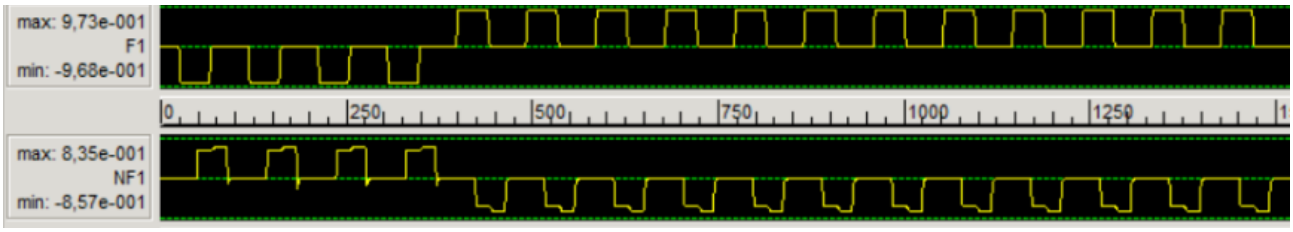


Рис.3.17

при T=40К (рис.3.18)

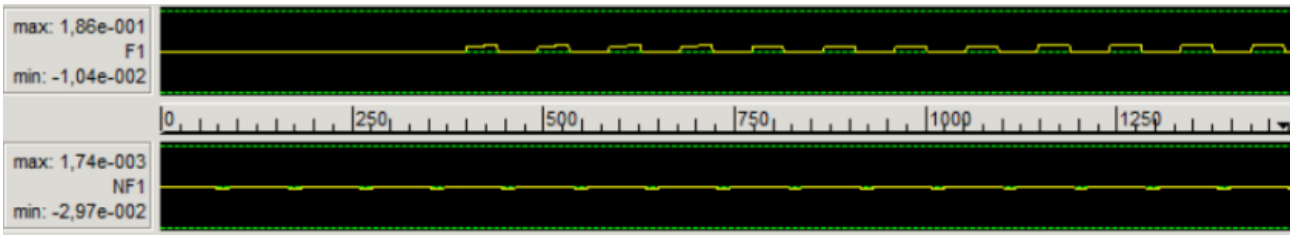


Рис.3.18

при T=70К(рис.3.19)

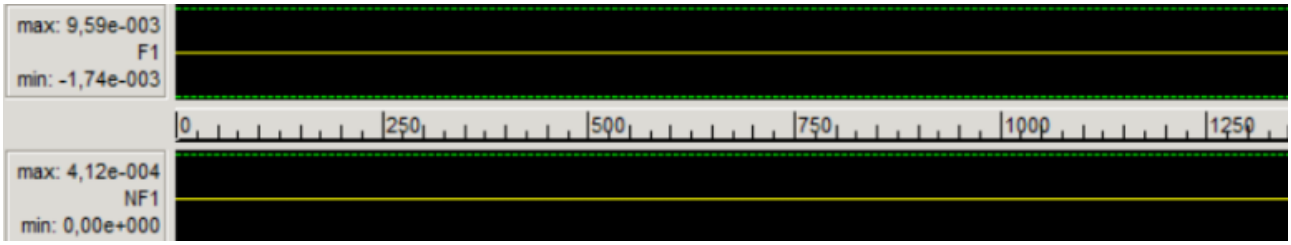


Рис.3.19

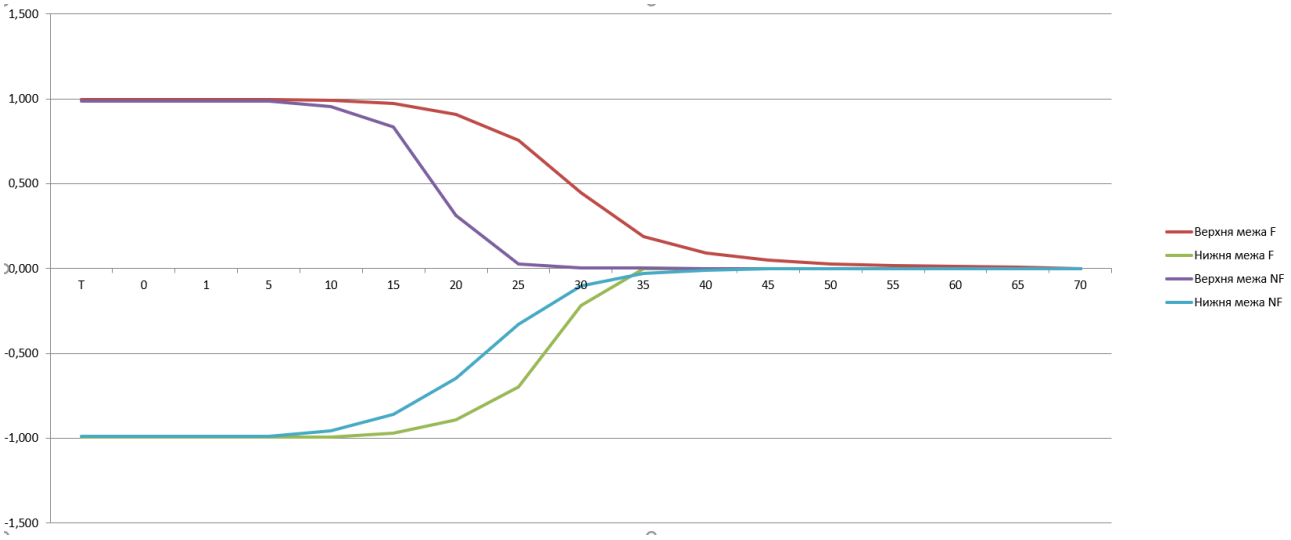


Рис.3.20. Графік залежності функцій від температури

5. Будуємо повну таблицю істинності для всіх проміжних та кінцевих виводів ПНЕС на КА.

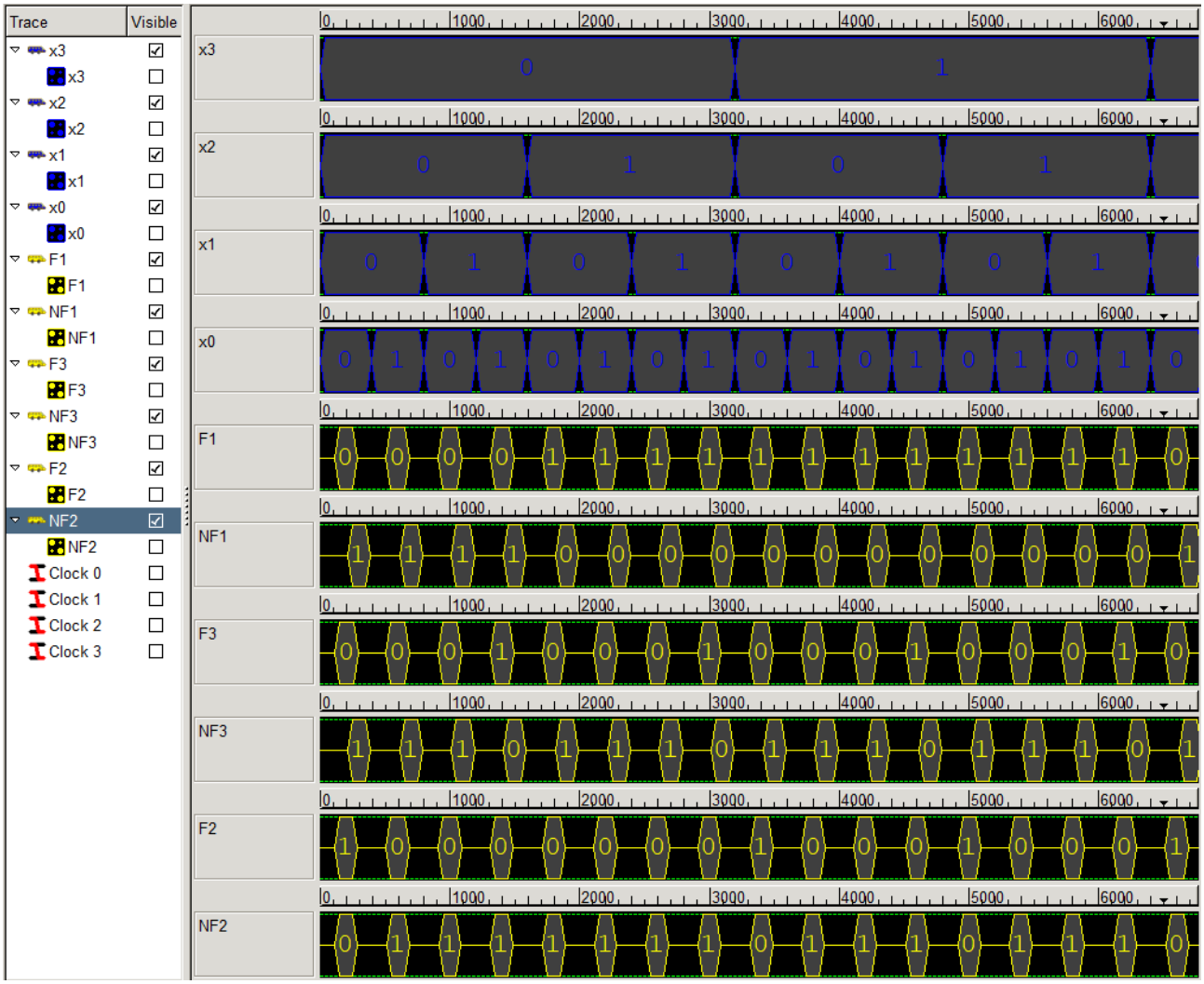


Рис.3.21. Діаграми вихідних логічних функцій

Таблиця 3.7

Таблиця істинності

№	x_3	x_2	x_1	x_0	f_1	$\overline{f_1}$	f_2	$\overline{f_2}$	f_3	$\overline{f_3}$
0	0	0	0	0	0	1	0	1	0	1
1	0	0	0	1	0	1	0	1	0	1
2	0	0	1	0	0	1	0	1	0	1
3	0	0	1	1	0	1	0	1	1	0
4	0	1	0	0	1	0	0	1	0	1
5	0	1	0	1	1	0	0	1	0	1
6	0	1	1	0	1	0	0	1	0	1
7	0	1	1	1	1	0	1	0	1	0
8	1	0	0	0	1	0	0	1	0	1
9	1	0	0	1	1	0	0	1	0	1
10	1	0	1	0	1	0	0	1	0	1
11	1	0	1	1	1	0	1	0	1	0
12	1	1	0	0	1	0	0	1	0	1
13	1	1	0	1	1	0	0	1	0	1
14	1	1	1	0	1	0	0	1	0	1
15	1	1	1	1	1	0	1	0	1	0

Постановка задачі №2

На мультиплексорі $(4 -> 1)$ запрограмувати виконання логічної функції двох або трьох аргументів згідно:

Таблиця3.8

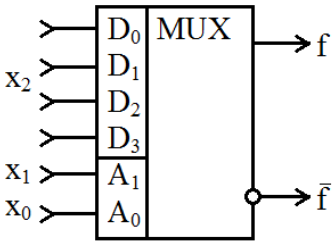


Рис.3.22

x_1	x_0	f	$\bar{f}$
0	0	D_0	$\bar{D}_0$
0	1	D_1	$\bar{D}_1$
1	0	D_2	$\bar{D}_2$
1	1	D_3	$\bar{D}_3$

Для реалізації логічних функцій двох аргументів на адресні входи A_1 та A_0 подавати бінарні аргументи x_1 та x_0, а інформаційні входи $D_3, \dots, D_0$ програмувати константами цих аргументів.

Для реалізації логічних функцій трьох аргументів третій аргумент x_2 подавати на один, декілька чи на всі інформаційні входи $D_3, \dots, D_0$, спростивши перед цим, по можливості, самі функції.

1. Використовуючи рівняння вихідної логічної функції мультиплексора $(4->1)$:

$$f = D_3 x_1 x_0 \vee D_2 x_1 \bar{x}_0 \vee D_1 \bar{x}_1 x_0 \vee D_0 \bar{x}_1 \bar{x}_0 \qquad (3.23)$$

довести справедливість програмування та створити повну таблицю істинності запрограмованої логічної функції.

2. Використовуючи САПР Quartus II, запрограмувати мультиплексор $(4->1)$ в якості ПЛІС та провести верифікацію проекту, яка включає:

- функціональне моделювання;

- часовий аналіз із заданими станами інформаційних входів;

- повну таблицю переліку станів схеми.

3. Створити наносхему мультиплексора $(4->1)$, та отримати діаграми вихідних функцій на базі квантових коміркових автоматів (КА) на проектному планшеті системи автоматизованого проектування (САПР) QCA Designer.

4. Дослідити залежність функцій від температури.

5. Розрахувати на САПР загальну кількість КА, їх розміри та відстань між їх центрами, діаметри квантових острівців і загальні розміри наносхеми з програмованими структурами.

Система автоматизованого проектування (САПР) Quartus II

Altera Corporation (NASDAQ: ALTR) — американська компанія, яка є одною найбільших виробників ПЛІС. Заснована у 1983 році.

В 1984 році компанією було випущено першу мікросхему програмованого логічного пристрою. Компанія проектує напівпровідникові прилади програмованої логіки (FPGA, CPLD, ASIC), розробляє програмні модулі та бібліотеки інтелектуальної власності для програмування ППЛ на мовах опису апаратних засобів (VHDL, Verilog), а також розробляє пакети програмного забезпечення для програмування ППЛ (пакет програм Quartus II).

Altera Quartus II - це середовище компанії Altera для програмування та розробки ПЛІС. За допомогою Quartus II можна аналізувати і синтезувати HDL конструкції, за допомогою чого розробник може складати свої проекти, виконувати часовий аналіз, тестувати RTL діаграми, імітує реакцію дизайну на різні подразники, і настроїти цільовий пристрій на програміста. До складу Quartus входить реалізація VHDL та Verilog для візуального редагування логічних схем та моделювання векторних сигналів, опису апаратного забезпечення.

Вирішення поставленої задачі №2 кваліфікаційної роботи

1. На мультиплексорі $(4{-}{>}1)$ запрограмувати виконання логічної функції:

$$x_2(x_1x_0 \vee x_1\overline{x_0} \vee \bar{x}_1x_0) \vee \overline{x_1x_0} \qquad (3.24)$$

Рівняння мультиплексора $(4{-}{>}1)$:

$$f = D_3x_1x_0 \vee D_2x_1\bar{x}_0 \vee D_1\bar{x}_1x_0 \vee D_0\bar{x}_1\bar{x}_0. \qquad (3.25)$$

Перетворена функція програмування мультиплексора:

$$f_{\text{пр}} = x_2(x_1x_0) \vee x_2(x_1\bar{x}_0) \vee x_2(\bar{x}_1x_0) \vee 1(\bar{x}_1\bar{x}_0). \qquad (3.26)$$

Таблиця 3.9

Таблиця програмування

x_1	x_0	D
0	0	$D_0 = 1$
0	1	$D_1 = x_2$
1	0	$D_2 = x_2$
1	1	$D_3 = x_2$

Таблиця 3.10

Таблиця істинності запрограмованої функції

x_2	x_1	$\overline{x_1}$	x_0	$\overline{x_0}$	f	$\bar{f}$
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	0	1	0	0	1
1	0	1	0	1	1	0
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	0	1	0	1	1

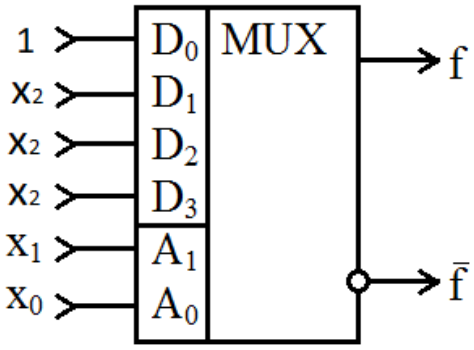


Рис.3.23. Програмування мультиплексора

2. Використовуючи САПР Quartus II, запрограмовуємо мультиплексор (4−> 1) в якості ПЛІС та проводимо верифікацію проекту

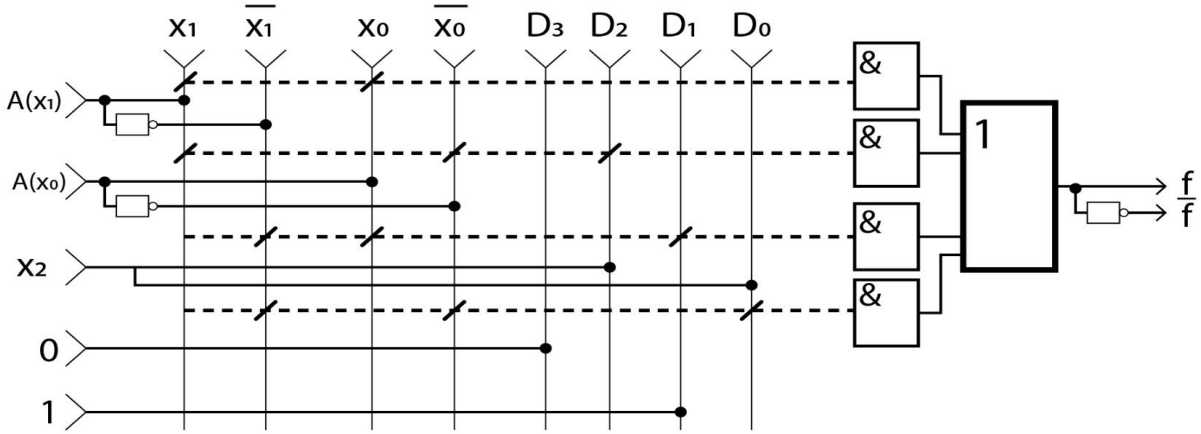


Рис.3.24. Реалізація функції на базових елементах «НІ», «І»,«АБО»

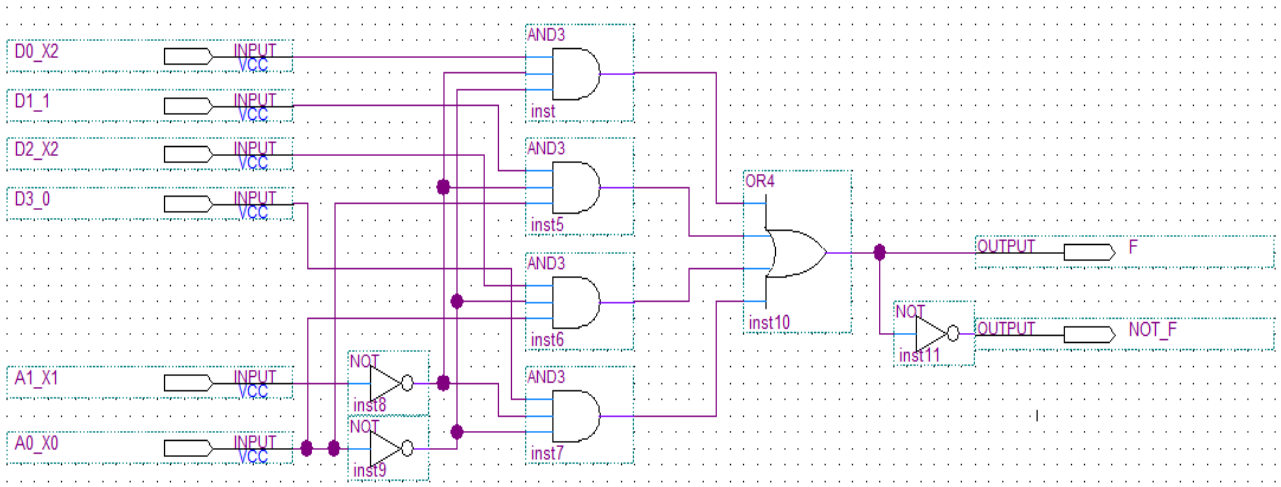
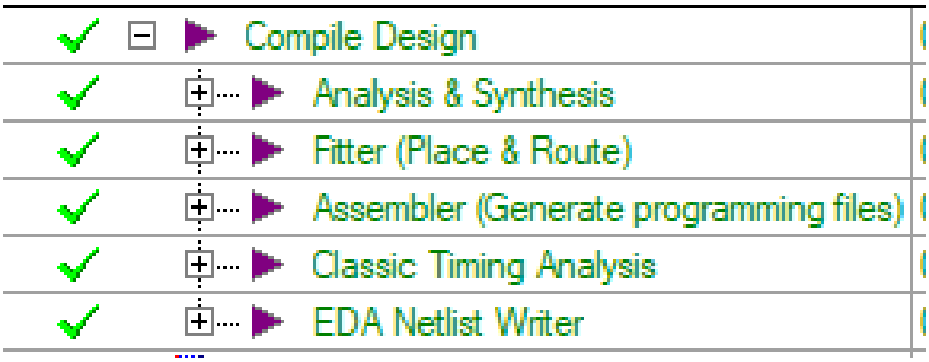


Рис.3.25.Функціональна схема запрограмованого мультиплексора в САПР Quartus II

Запуск проекту:



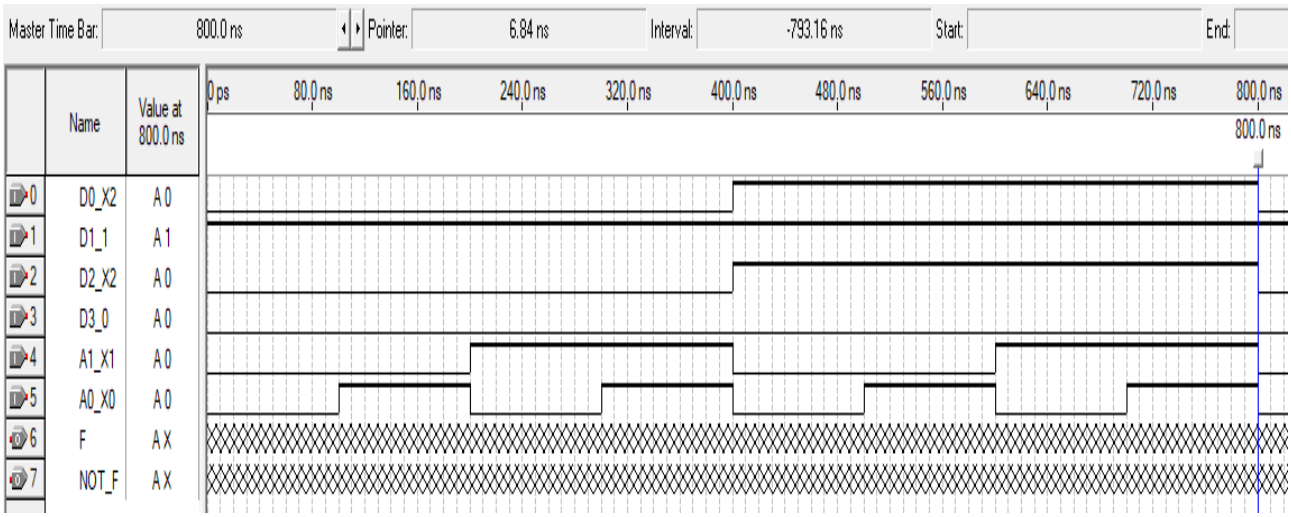


Рис.3.26.Часовий аналіз із заданими станами інформаційних входів

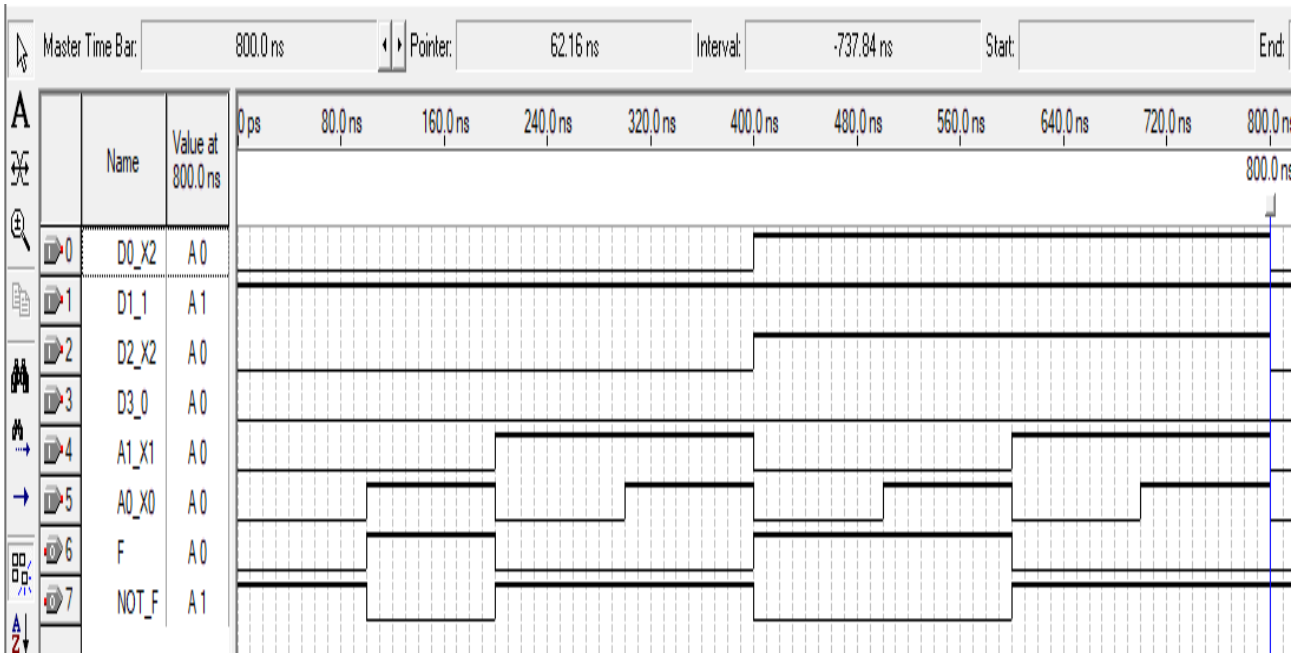


Рис.3.27.Часовий аналіз станів схеми та затримки сигналів

3. Створюємо наносхему мультиплексора $(4 -> 1)$, та отримуємо діаграми вихідних функцій на базі квантових коміркових автоматів (КА) на проектному планшеті системи автоматизованого проектування (САПР) QCA Designer.

Моделювання функції запрограмованого мультиплексора на можаритарних наноелементах:

$$f = x_2(x_1\overline{x_0} \vee \overline{x_1 x_0}) \vee \bar{x}_1 x_0 \qquad (3.27)$$

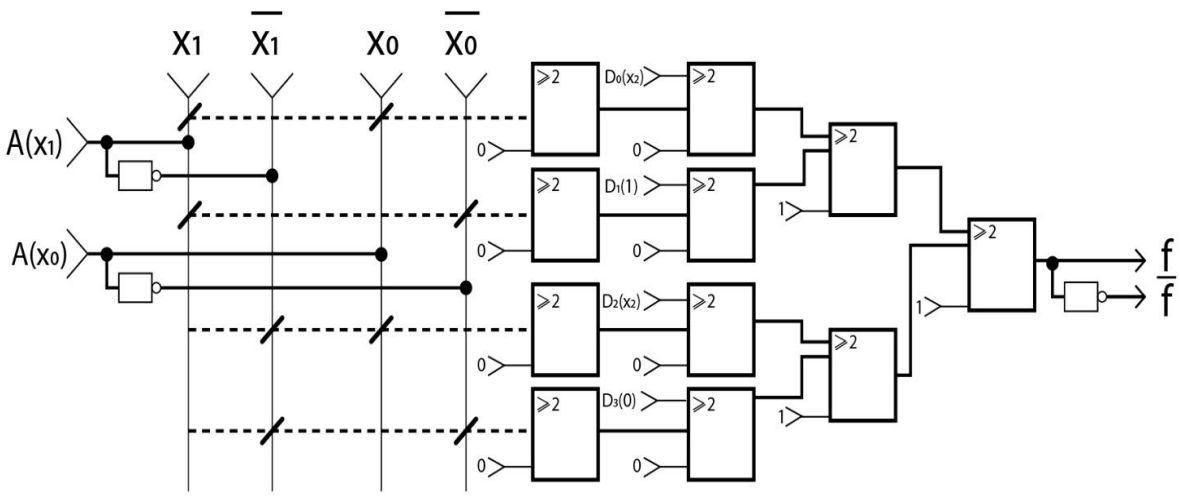


Рис.3.27

Наносхема запрограмованого мультиплексора (4->1) на базі квантових автоматів,яка реалізує задану функцію:

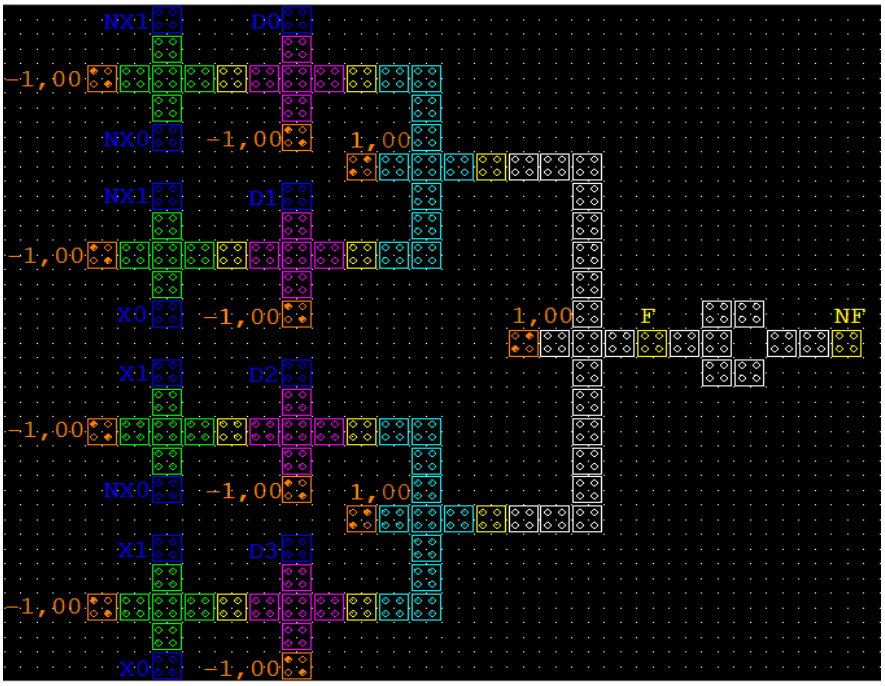


Рис.3.28. Змодельована схема

```
Selection extents: (471,00,271,00)[98,00x58,00] = 5684,00 nm^2 = 0,01 um^2 Objects selected: 8
Simulation found 12 inputs 12 outputs 124 total cells
Starting initialization
Total initialization time: 0 s
Starting Simulation
Total simulation time: 10 s
```

Часовий аналіз сигналів запрограмованої наносхеми:

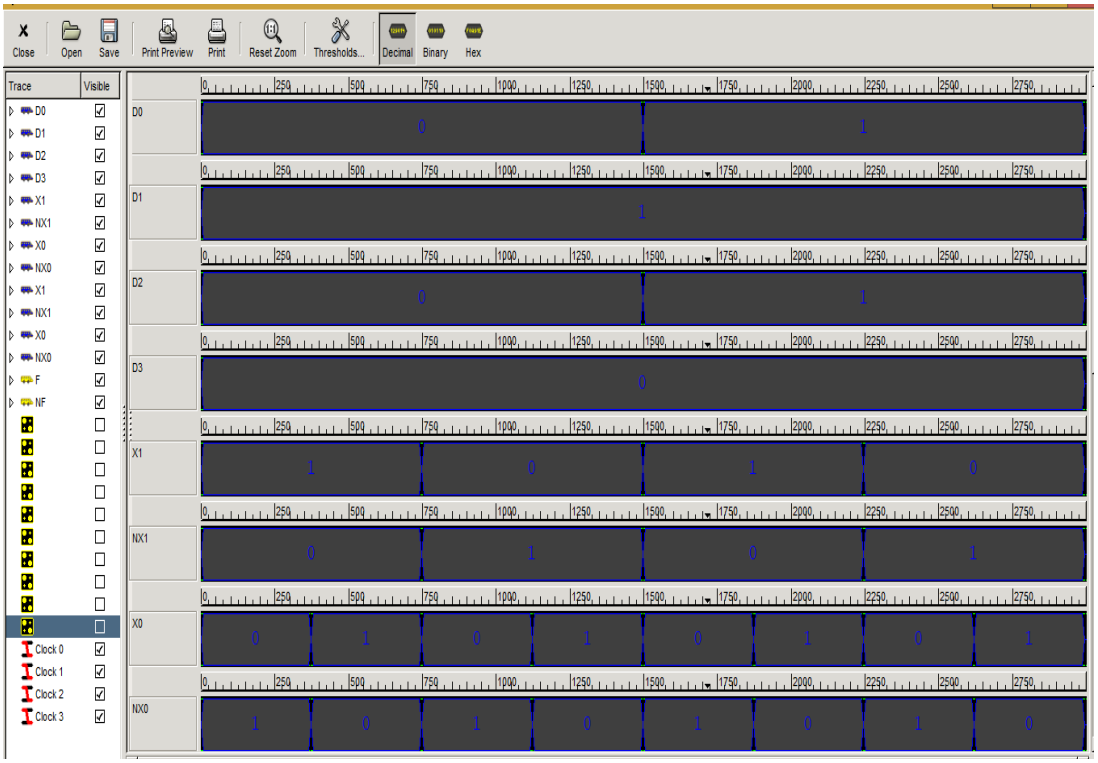


Рис.3.29

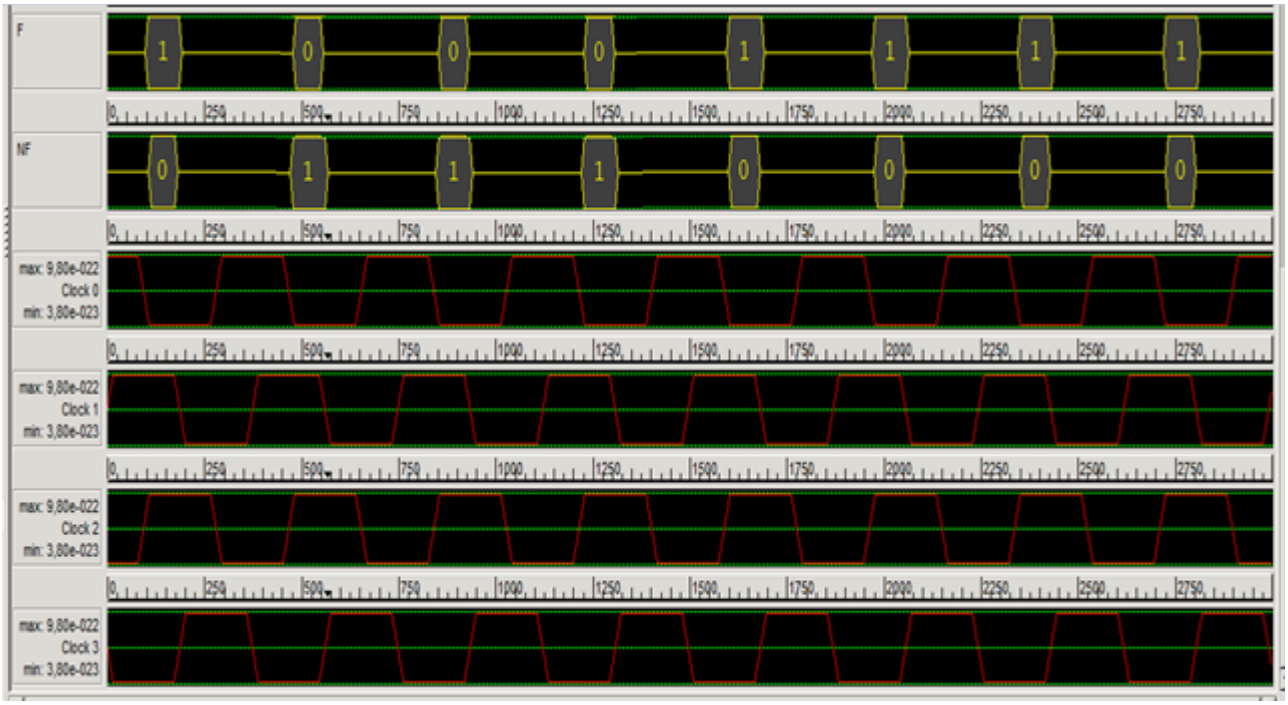


Рис.3.30

4 . Досліджуємо залежність функцій від температури .

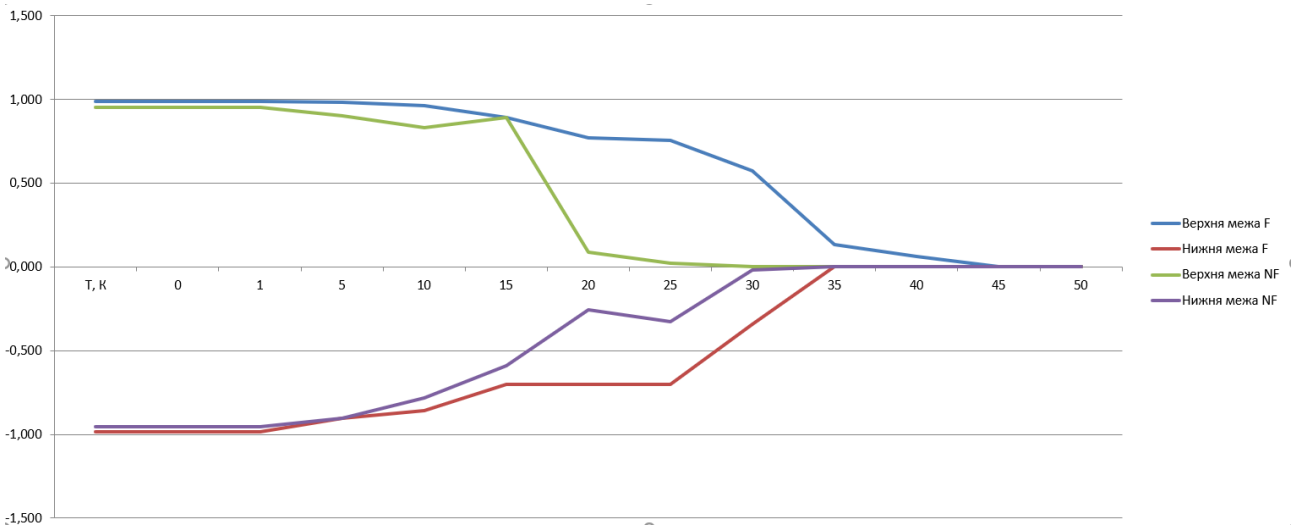


Рис.3.31. Графік залежності фукції від температури

5. Розраховуємо на САПР загальну кількість КА, їх розміри та відстань між їх центрами, діаметри квантових острівців і загальні розміри наносхеми з програмованими структурами.

- Загальна кількість квантових комірок: 124, у яких 12 входів і 12 виходів
- Розмір квантової комірки: (18x18)нм
- Відстань між центрами КА: 20нм
- Діаметр квантового острівця: 5нм
- Загальні розміри запрограмованої наносхеми: (454,8x362,81) нм = 165005,988 нм^2 = 0,165 мкм^2

3.3. Висновок до третього розділу

Комп'ютерне моделювання першої задачі здійснено за допомогою САПР QCA Designer. Часові діаграми запрограмованих функцій співпали з таблицею істиності.

Під час виконання другої задачі було запрограмовано мультиплексор на виконання необхідної логічної функції. Мультиплексор був запрограмований аналітично, за допомогою САПР Quartus II, та за допомогою САПР QCA Designer. Часові діаграми запрограмованих мультиплексорів в різних середовищах співпали з таблицею істиності.

Щонайекономічніші наносистеми з програмованою логікою споживають лише (3.8-9.8)e-22 Дж енергії в колах синхронізації та живлення (рис.3.15).

Порівнюючи температурну залежність двох схем, бачимо, що програмована функція першої схеми більш надійна до працездатності. Це пов'язано з кількістю КА.

Термін "наноелектроніка" відображає перехід до компонентів розмірами в нанометровій області, та його можна логічно пов'язати з терміном "мікроелектроніка".

Принципово новою особливістю наноелектроніки є те, що для компонентів таких розмірів починають переважати квантові ефекти. Відриваються нові перспективи їх використання, появляються нові номенклатури характеристик. Електроніка, яка застосовує квантові ефекти – наногетероструктурна електроніка.

Одним з найперспективніших компонентів цифрової схемотехніки є програмовано логічні інтегральні схеми (ПЛІС). ПЛІС представляє собою кристал, яка вміщує велику кількість базових ЛЕ. На початку ці елементи не з'єднані між собою, які за допомогою електронних ключі з'єднануються, перетворюючі ці компоненти в електричну схему. Управляти цими ключами можна за допомогою спеціальної пам'яті, до якої записується код конфігурації цифрової схеми. Таким чином, записавши в пам'ять ПЛІС певні коди, можна зібрати цифровий пристрій будь-якого ступеня складності, це залежить від кількості елементів на кристалі і параметрів ПЛІС. В ПЛІС можна організувати алгоритм цифрової обробки, чого не можна зробити в мікропроцесорах. Швидкодія при цьому стрімко зростає. Вищість технологій проектування пристроїв на ПЛІС:

• малий час виговлення схеми, для цього потрібно тільки внести в пам'ять ПЛІС код конфігурацій;

• немає необхідності розробляти та виготовляти складних плат;

• під час зміни коду конфігурації схеми в пам’ять, проходить швидке перетворення однієї конфігурації цифрової схеми в другу;

• не потребує складного технологічного виробництва.

ВИСНОВОК

Відкриття нової сторінки в історії теперішньої мікро- та наноелектроніки та обчислювального обладнання відбулося завдяки програмуванню користувачем мікро- та наносхем. Завдяки цьому було створено ВІС, які застосовуються для виконання спеціалізованих завдань виробами промислової електроніки зі всіма випливаючими з цього схвальними результатами: масове виготовлення, низька ціна мікросхем, скорочення часу виготовлення і виходу на ринок продукції. На сьогоднішній час існують чотири класи програм мікро- та наносхем.

В кваліфікаційній роботі досліджено та проаналізовано архітектурно структурну організацію ПЛІС. Розглянуті сучасні напрямки розвитку ПЛІС-технологій. Окреслено актуальність та проблеми створення обчислювальних пристроїв на їхній основі. Проаналізовано засоби та методи, архітектурно-структурні рішення для вирішення проблеми моніторингу об’єктів критичного застосування на принципі використання програмовано логічних інтегральних схем для здійснення таких систем.

Розглянуті базові ознаки класифікації мікро- та наносхем з програмованими характеристиками, запропоновані класи за архітектурою. Будова ПЛІС ускладнилася завдяки збільшення рівня інтеграції ПЛІС, появилася архітектура, яка в тій чи іншій мірі поєднує переваги CPLD і FPGA. До числа мікро- та наносхем з такою архітектурою, можна віднести сімейство Flexible Logic Elementmatri X.

Було змодельовано наносхему на КА завдяки САПР QCA Designer. Значення таблиці істинності співпали з часовими діаграмами запрограмованих функцій.

У наступні роки напівпровідникові елементи великих інтегральних схем досягнуть квантово-технологічних обмежень і не зможуть відповідати наростаючим вимогам ефективності обчислювального обладнання. З огляду на це, активно розробляються нові нанотехнології, які б забезпечили суттєво вищу ефективність. Однією з таких розробок є квантові коміркові автомати і створенні на їх основі НСКС.

Термогенерація носіїв заряда при криогенних умовах (0...50К) суттєво обмежує надійність працездатності наносхем. Генеровані електрони викликають спотворенн імпульсних характеристик одноелектронних наносхем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Матвієнко М.П. Комп'ютерна логіка. Підручник. К.: Ліра, 2018 - 324с.
2. Сергієнко А.М. Багатопроцесорні системи на ПЛІС. Підручник. К.: Політехнік, 2017.- 127с.
3. Рябенький В.М., Жуйков В.Я., Ямненко Ю.С. Схемотехніка, т.1,2,3. Підручники. К.: Вища школа. 2014-2018
4. Sergienko A.M. Computer Network Engineering. Навчальний посібник англійською мовою. К.: "Корнейчук ", 2017.- 80с.
5. Melnik O.S., Yurchyk I.l. Nanodevices with Programmable Logic. // Int. Science Journal "Electronic and Control Systems". 2019, v.4, N62, p.47-51. Режим доступу: https://jrnl.nau.edu.ua/index php/ES4/article/download/14383/20519
6. Говорущенко Т.О. Типові цифрові схеми комп’ютерів.[Текст] –Харків, 2015.- стор. 12.
7. Мельник О.С., Миколушко А.М. Репрограмовані мультиплексорні наносхеми. // Математичне моделювання в техніці та технологіях. N8(1333), 2019. с.224-231. Режим доступу:http://repository.kpi.kharkov.ua/bitstream/KhPI-Press/42076/1/vestnik_KhPI_2019_8_Melnyk_Reprohramovani.pdf
8. Методи обробки даних на ПЛІС.[Електронний ресурс].–2015. Режим доступу: http://ela.kpi.ua/bitstream/123456789/19096/1/Klymenko_aref.pdf
9. Характеристики ПЛІС. [Електронний ресурс].-2015. Режим доступу: https://msn.khnu.km.ua/pluginfile.php/147595/mod_resource/content/2/Lekz10.pdf
10. Spartan-3A FPGA EK Family Data Sheet. [Електронний ресурс].-2015. Режим доступу: http://datasheet.elcodis.com/pdf2/102/36/1023636/xc3s1400an.pdf
11. Intel FPGAs. [Електронний ресурс].-2010. Режим доступу: https://www.intel.com/content/www/us/en/products/programmable/fpga.html
12. Serias FPGAs Xilinx: Overview. [Електронний ресурс].-2010. Режим доступу: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf
13. Мельник О.С. Програмовані мікро- та наносистеми. Підручник. [Електронний

ресурс]. Режим доступу: http://kafelec.nau.edu.ua/Materialu/MELNIK_%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BE%D0%B2%D0%B0%D0%BD%D1%96%20%D0%BC%D1%96%D0%BA%D1%80%D0%BE-%20%D1%82%D0%B0%20%D0%BD%D0%B0%D0%BD%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8%20%D0%B2%20%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D1%96%D1%86%D1%96.pdf

14. Тишкова І.О. Температурні обмеження працездатності програмованих наносхем // Політ. Сучасні проблеми науки : тези доповідей XXI Міжнародної науково-практичної конференції здобувачів вищої освіти і молодих учених . – Національний авіаційний університет. Режим доступу: https://er.nau.edu.ua/bitstream/NAU/50170/1/Tishkova.pdf